AD NUMBER

AD469269

NEW LIMITATION CHANGE

TO

Approved for public release, distribution unlimited

FROM

Distribution authorized to U.S. Gov't. agencies and their contractors; Administrative/Operational Use; Jul 1965. Other requests shall be referred to Director, Rome Air Development Center, Attn: RTD, Griffiss AFB, NY.

AUTHORITY

RADC ltr, 31 May 1966

SECURITY MARKING

The classified or limited status of this report applies to each page, unless otherwise marked.

Separate page printouts MUST be marked accordingly.

THIS DOCUMENT CONTAINS INFORMATION AFFECTING THE NATIONAL DEFENSE OF THE UNITED STATES WITHIN THE MEANING OF THE ESPIONAGE LAWS, TITLE 18, U.S.C., SECTIONS 793 AND 794. THE TRANSMISSION OR THE REVELATION OF ITS CONTENTS IN ANY MANNER TO AN UNAUTHORIZED PERSON IS PROHIBITED BY LAW.

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

RADC-TR-65-189, Vol II
Final Report

DESIGN OF RELIABILITY CENTRAL DATA MANAGEMENT SUBSYSTEM

Dr. J. Sable
et al
(Auerbach Corp)

TECHNICAL REPORT NO. RADC-TR-65-189
July 1965

Information Processing Branch
Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

DESIGN OF RELIABILITY CENTRAL DATA MANAGEMENT SUBSYSTEM

Dr. J. Sable
et al

AFLC, GAFB, N.Y., 26 Aug 65-188

VOLUME II

TABLE OF CONTENTS

Paragraph	Title	Page
	SECTION I. INTRODUCTION	
1.1	GENERAL	1-1
1.2	ORGANIZATION OF VOLUME II	1-2
1.3	RECENT DEVELOPMENTS TOWARD GENERALIZED DATA MANAGEMENT	1-3
1.4	SUMMARY OF CONCLUSIONS	1-8
	SECTION II. DATA MANAGEMENT SYSTEM COMPARISON CHART	
2.1	A CLASS OF DATA PROCESSING SYSTEMS	2-1
2.2	COMPONENTS OF THE DATA MANAGEMENT SYSTEM	2-1
2.3	SYSTEM CONCEPTS AND TERMINOLOGY	2-3
2.4	COMPARISON CHART	2-8
	SECTION III. DESCRIPTION OF THE RELIABILITY CENTRAL DATA MANAGEMENT SUBSYSTEM	
3.1	OBJECTIVES OF THE RELIABILITY CENTRAL	3-1
3.2	CHARACTERISTICS OF THE DATA BASE	3-3
3.3	DESIGN OBJECTIVES OF THE DATA MANAGEMENT SUBSYSTEM	3-4
3.4	CHARACTERISTICS OF THE DATA MANAGEMENT SUBSYSTEM	3-12
	SECTION IV. SYSTEM DESCRIPTIONS AND COMPARISONS	
4.1	GENERAL	4-1
4.2	RETRIEVAL COMMAND ORIENTED LANGUAGE (RECOL)	4-3
4.3	ACSI-MATIC COLLATION SYSTEM	4-10
4.4	NAVAL AVIATION SUPPLY OFFICE SYSTEM (ASO)	4-22
4.5	INFORMATION PROCESSING SYSTEM (IPS)	4-29
4.6	HQ, USAF COMMAND & CONTROL SYSTEM (473L)	4-40
4.7	COMPILE ON-LINE AND GO SYSTEM (COLINGO)	4-54
4.8	LANGUAGE USED TO COMMUNICATE INFORMATION SYSTEM DESIGN (LUCID)	4-65
4.9	ADVANCED MANAGEMENT SYSTEM (ADAM)	4-74
	SECTION V. SUMMARY OF COMPARISONS AND CONCLUSION	
5.1	PROCESSING CAPABILITY	5-1
5.2	USER LANGUAGES	5-3
5.3	FILE STRUCTURE	5-5
5.4	PROGRAMMER INTERFACE	5-7
5.5	RESPONSE TIME	5-9
5.6	CONCLUSIONS	5-11

SECTION I. INTRODUCTION

1.1 GENERAL

AUERBACH Corporation has been retained by Rome Air Development Center* to design a file structure and an information storage and retrieval system for the planned U.S. Air Force Reliability Central. Volume I of this Final Report presents a functional description of the system, called the Reliability Central Data Management Subsystem (RCDMS). A second basic task under the contract is to show:

(1) how the design has met the specific operational objectives of the Reliability Central,

(2) how the design has drawn on the available state of the art, and

(3) where the design philosophy and design techniques advance the state of automated data management and data analysis technology.

Volume II of this Final Report is presented in response to the second contract requirement. Volume II compares the design of the Reliability Central Data Management Subsystem (RCDMS) with other major data management systems in existence or currently under development.

It is difficult to present a comparative report of many systems in one technical area, without giving the erroneous impression that judgment is being passed on the systems under consideration. There is no such intent in this report. In order to make comparative analysis meaningful, the functions, design features, and operational capabilities being designed into the RCDMS are compared separately with each of the other systems. These comparisons should be treated as comparisons of facts, not as judgments as to whether the

*Under Contract AF 30(602)-3433

features planned for the RCDMS should have been present in the other systems. Each of the systems discussed has been designed in response to a different set of objectives: some as experimental systems, some in response to specialized requirements, some as technique-proving vehicles. They have all been designed at different points in the evolution of data management technology. During the analysis of the systems, AUERBACH Corporation was impressed with the extent to which the designers met their objectives while advancing the state of the art. Therefore, the comparison of features present in the planned RCDMS design against the features present in all the other systems must not be interpreted as a criticism of these systems. The set of objectives for RCDMS are quite demanding since it has to be a user-oriented production system. Therefore, it was necessary to draw on all that is best in the technology. The design of the system consolidates concepts, design techniques, and approaches to data management which were often originally developed by the designers of the systems with which it is now being compared.

Furthermore, the RCDMS design was initiated much later than the systems with which it is being compared. RCDMS is now in its design stage, even though concepts, techniques, and operational objectives are firmly defined. The designers were in an excellent position to draw on the experience of others and to take the experience a step forward, instead of merely duplicating the past.

1.2 ORGANIZATION OF VOLUME II

This Section (Section I) contains general information and background on the development of data management systems, and a summary of report conclusions.

Section II provides a consolidated chart summarizing the system comparisons from which the conclusions are drawn.

Section III contains a description of the objectives and design of the Reliability Central Data Management Subsystem (RCDMS). (Section III has been extracted from Sections III and IV of Volume I in order to add continuity to Volume II).

Detailed comparisons between the RCDMS and each of the other systems are in Section IV.

A comprehensive discussion of all comparisons and conclusions will be found in Section V.

1.3 RECENT DEVELOPMENTS TOWARD GENERALIZED DATA MANAGEMENT

The RCDMS is being designed to perform as a general-purpose data management system, in order to meet economically the anticipated growth and changing analytical needs of the Reliability Central Program. Data management systems are an outgrowth of the dynamic history of data processing development, and the generalized data management system approach now represents the most advanced method of handling and providing for the use and analysis of large volumes of data. Recent developments leading to the concepts and development of general-purpose data management systems are outlined below.

1.3.1 Early Query Systems

Whereas procedure-oriented languages had solved many file maintenance and programming problems, they had not solved retrieval problems. Users wanted to retrieve data in several different ways, not just by the name of the data but by the characteristics of the data — the values of data items. In addition, program designers were aware of the user demands that data often had to be processed only when it satisfied certain logical conditions. Prior to the introduction of query languages and systems, programs had to be rewritten each time the set of logical conditions was changed. An

effort was made to apply the generalized program approach to the handling of queries. The goal was to prepare a single program responsive to a flexible language which expressed complex logical conditions by means of Boolean operators. One of the first such languages called QUERY was published in late 1960. QUERY permitted flexible English-like requests to be stated for retrieval processing. Although never implemented, QUERY was influential in the development of other query languages.

Other efforts in the early 1960's included the original language developed for ASCI-MATIC, which used a kind of basic English, and initially permitted the AND operator only. It was one of the first languages developed which permitted the use of magnetic disc storage, and was designed with a multi-list structure with link addresses appearing in a directory rather than in the data records. BASEBALL was another generalized query system designed for a special purpose. BASEBALL used the generalized program approach to accept statements very close to normal English, but was restricted to the special environment imposed by the words and data involved with the game of baseball. BASEBALL was a limited experiment to show what language could do to help the interplay between man and machine.

The first general-purpose query system was developed during 1961 and 1962 at the David Taylor Model Basin. Called TUFF-TUG, it was the predecessor of IPS (Information Processing System). Other examples of languages permitting general queries to any kind of file are RECOL and GIRLS. Different users could connect their own file to those systems; they could present their file description to the system, and could then process a query without any additional programming. The languages permitted the unrestricted nesting of logical conditions, as well as the use of relational operators such as "equal to," "greater than," and the like. The languages did not permit algebraic statements. Since no "if" statement was available in the user language, selective print formatting and selective output ordering were not available.

1.3.2 File Management Systems

A new level of data processing power and generalization was conceived when system designers realized that many files could be handled if the system were given a directory describing the format of the files and where the files were located. Such systems as IPS, COLINGO, and 473L were developed with the notion that once the files were defined, the system could perform both maintenance and retrieval functions on many different files on an automatic basis. Thus was born the concept of file management. The systems possessed an extended language and system capability, as well. For instance, algebraic and functional operators were permitted in search questions. Complex file maintenance tasks could be performed using Boolean operators in the language to control the tasks. For example, complex output formats and arrangements could be achieved with the extensive display capability of the Air Force 473L system. The IPS system used only magnetic tape. COLINGO had its programs on magnetic disc but its files were on tape. The 473L system employed only discs, but the system did not utilize their random access potential. Consequently, it can be said that the early file management systems were essentially tape-oriented.

1.3.3 Data Management Systems

The notion has emerged that the functions of data maintenance and retrieval (as discussed in the previous paragraph) are not ends in themselves. They must be combined with other user functions in order to realize their full potential. Queries, for instance, need not show only the status of the data; their output can be used as input to complex computations for any number of analytical purposes. Data management systems attempt to make it easy for the untrained user to link many tasks together for his use, without having to reprogram any part of the system. Such systems have within their framework a large repertoire of task-oriented programs which can be linked

together in order to perform a wide variety of specialized jobs. The systems employ command languages with which the user states the parameters and the sequence of the programs to be run. The systems contain large, generalized executive routines which link the task-oriented programs. In this manner, such data management systems as ADAM and LUCID have incorporated many specific tasks into a flexible, generalized and general-purpose framework which can change responses from one user to the next with minimum change to the system. However, they are not entirely production-oriented because they are designed for a specific task — that of testing system designs. Thus far, no truly general-purpose data management systems have been reported.

Random access storage devices and deep indexing are changing the nature of data management systems. The use of random access devices permits data to be extracted and combined from many files, thereby blurring the distinction between files, and causing all data to become part of a large, single entity called a data base. The combination of random access with the kind of indexing that goes deeply into the content of the data adds power to data management systems. Communication between man and the computer is increased as never before, because man is able to call up large volumes of information and have it processed according to his wants while he is on-line with the computer.

1.3.4 RCDMS as a General Data Management System

RCDMS treats the design of a data management system as a unified whole, in which the accessibility of data to the user, and the ease of its subsequent manipulation and analysis in a production rather than an experimental environment are paramount design considerations. The design also takes advantage of the resources, both software and hardware, of the RADC EDP complex, in which the design is to be initially implemented. The key distinguishing characteristics of the design are as follows:

(1) Provision of a data description language for entering into the system data definitions which subsequently govern the input and structuring of data which the definition describes. This permits the user to enter into the system any data structured in response to the user's anticipated need of the data.

(2) A command language which permits the user to describe jobs composed of one or more pre-stored program tasks, by stating data parameters, and identifying the sequence in which the tasks are to be executed. The command language further permits the user to call for the execution of any job by naming it and supplying data parameters to be bound at job run time.

(3) Provisions for extracting from the data base any data required for the execution of user-defined jobs by means of complex Boolean expressions using A), OR, >,<, =, range, and functional operators.

(4) Provision of a directory system with in-depth indexing capability which permits the retrieval of any indexed data from anywhere in the data base without the need to search the data itself.

(5) Structuring the data base logically and independently of the physical location of the data. This makes the dynamic maintenance of a large data base possible. The changes in logical structure of the data base, as described by the directories, need not require physical movement of the data itself. Correspondingly, physical relocation of the data need not involve the changes in the logical structure of the data, as given by the directories.

(6) The structure of the data base lends itself directly to the application of a Boolean search strategy system which optimizes retrieval. The search strategy system provides a flexible interface for a query system with a high level user language.

1.4 SUMMARY OF CONCLUSIONS

(1) The RCDMS design has drawn heavily on the advances in the state of the art already achieved in the field of data retrieval and data management.

(2) The basic contribution made by RCDMS is that it has drawn together into one system many major aspects currently associated with data management. This consolidation of system features should make RCDMS more responsive to user needs and more of a general-purpose production system than the other systems surveyed.

(3) The RCDMS design helps to divorce the user from the complexities of the system. RCDMS deals with the data base more as a single entity than as a collection of files. By permitting nesting and cross referencing of data, by indexing data automatically, by keeping statistics on data usage, and by providing the user with flexible and easily-used languages, the RCDMS design permits the user to alter his data easily when his needs change. More rigid systems require the user to plan ahead more carefully, and to become more familiar with the details of the system so he will be sure his plans can be implemented.

(4) Several original contributions to data management systems are contained in the RCDMS design:

(a) RCDMS makes full use of the potential of random access storage by keeping indexing data efficiently stored in the directory and by using an optimized search strategy which accesses only pertinent data from peripheral storage devices.

(b) RCDMS separates the logical address of data from the physical addresses so that minimal directory maintenance is required when data is moved from one physical location to another, and so that actual data movement is minimized when the logical structure of the data base is modified.

(c) Of all the systems surveyed, RCDMS appears to have the most flexible user-oriented method for having jobs specified by the user. Further, the jobs can be pre-defined and stored before run time. The program library directory contains sufficient parametric data to permit automatic linking of programs as specified by the job description.

(d) RCDMS will keep track of and analyze the frequency of access to various kinds of data, and will automatically notify operating personnel of data structure changes which might be more efficient. Eventually it is hoped that these changes can be made automatically within the system. A further by-product of the statistical analysis is that RCDMS will automatically report the level of accessibility best suited to a segment of data and will transmit this information to the RADC Local Control and Executive Control Programs for their discretionary use when the data segment is to be stored.

SECTION II. DATA MANAGEMENT SYSTEM COMPARISON CHART

2.1 A CLASS OF DATA PROCESSING SYSTEMS

The specific class of automatic data processing systems discussed in this report has been called "data management systems." In a data management system (DMS), the software environment of a data processing center provides a tool to a staff of users, programmers, and operators for maintaining and interrogating a large data base. The data processing center, on the other hand, exists in a problem environment that places stringent functional requirements upon it — requirements for the rapid and accurate assimilation of new data for changes in the data structure, for interrogation of the data contents, and for analytic results based on data content.

2.2 COMPONENTS OF THE DATA MANAGEMENT SYSTEM

A DMS has a number of elements which can be identified and are discussed below. The capability of a DMS as a whole depends on how these elements have been implemented, their flexibility, capability, etc. It is the purpose of this report to attempt to catalog and evaluate these elements in a number of systems being designed, implemented, and operated. The system comparison chart included at the end of this section is a summary of the evaluation of nine data management systems.

The following elements can be identified in a DMS.

(1) User Language. The user language is the vehicle for communicating job requests, queries, and, in some systems, new data structures to the system.

(2) File Structure. The data base is the main resource of the DMS. The data base has logical structure and content. The logical structure is represented by the named elements of the data base and their interrelationships. The content is the data itself, mapped onto the physical storage media of the data processing center. In some systems this mapping is explicit, in the form of directories and indexes. In other systems the mapping is implicitly contained in the program logic of the data management programs.

(3) Processing Capability. The DMS contains programs which search the data base, maintain the data and its directories, format the output, and manage the specialized jobs of the center.

(4) Programmer Interface. Libraries of macros and programs, and procedural language compilers are provided for the programmer.

(5) Response Time. A major result of designing a DMS as an integrated whole is an increase in the responsiveness to the user, usually via direct on-line inquiry consoles. This is accomplished by providing various levels of priority and accessibility and using search strategies that take advantage of the data structures, directories, and random access storage devices in the system.

2.3 SYSTEM CONCEPTS AND TERMINOLOGY

Since there are no terminology standards in the data management field, and since many concepts and philosophies of data management are still in their formative phase, some terminological obstacles have arisen in the description of these data management systems.

In general, the terminology of the designers has been preserved in describing the various systems in this report. In some cases, two groups use similar terms to describe different things and, in other cases, new terms have been given familiar concepts. An effort has been made to guard against confusion through the use of footnotes and parenthetical remarks when these situations arise. The following remarks are designed to serve as further guideposts in reading and evaluating this report.

2.3.1 User Language

The term "user" itself is used ambiguously, due to the different orientations of the various systems. In RCDMS, COLINGO, and the HQ,USAF system, for example, the user is the operational staff member who is querying the data base or running a job. However, in LUCID or ADAM the "user" is the programmer or system designer who is modeling or constructing a system.

The user language is sometimes characterized as a Query Language and sometimes as a Job Request Language. The following approaches in the designs of the user language can be observed:

(1) The Query Imbedded in the Job Request Language. This is the approach adopted by RCDMS. The Job Request Language has components which can be used to define data structure, modify data, enter command vocabulary, store data descriptions, and run jobs which bind the parameters of job descriptions. In this view, a query is a particular job which assigns parameters to a query routine.

(2) Job Requests Imbedded in the Query. In this approach, the query is seen as the more generic element, and specific system actions can be requested in what is called a query. This is the approach used in COLINGO and the HQ, USAF systems.

(3) Separate Job Request and Query Languages. The treatment of queries and jobs as two separate entities is evident in the IPS system which treats the query function and the data maintenance function in two separate systems: the IRS (Information Retrieval System) and the FMS (File Maintenance System).

(4) User-Constructed Languages. Finally, there are systems such as ADAM which treat the relationship of the Job Request Language and the Query Language as something to be modelled by the designer, with no a priori languages built into the system.

It is also illuminating to categorize user languages along two additional dimensions:

(1) Flexibility with Regard to Language Restrictions. In the dimension of language flexibility a spectrum of (query) languages exists. They range from fixed, push-button initiated responses (which, for example, associate a given fixed status report with a console push-button) to a natural "near-English" language which is entered via a keyboard and is capable of eliciting a wide range of responses. An example of a language close to the fixed end of the spectrum is the ASO query system. COLINGO, on the other hand, lies near the multiple-response end.

(2) Process versus Goal Orientation. With respect to the degree of process orientation, a spectrum of languages exists. They range from procedure-oriented "algebraic" compiler languages such as JOVIAL to completely process-independent, goal-oriented languages such as RECOL.

2.3.2 Data Structure Terminology

One aspect in all data management systems (and essential to their understanding) is the clear distinction between data content and data structure. The data content is referred to as the data base, or as the set of data values that comprise the declarative statements in the system which are concerned with the outside world (sometimes referred to as the "real world"). The data base makes up what is usually the majority (but by no means all) of the data in the data management system. A large part of the data in the system concerns the vocabulary, structure, definition, and location of data items in the data base. This data descriptive data (data about data) is located in the system directory (or directories). The data base is sometimes called the system files, and a directory is sometimes referred to as a system table, glossary, thesaurus, index, or (as in ADAM) a roll.

In discussing data structure, it is important to distinguish clearly between physical structure and logical structure. (A failure to make this distinction clear usually results in a good part of the confusion and "semantic noise" in this area.) Physical structure refers to the disposition of data on the storage media. This covers the location and disposition of physical data elements usually connotated by such words as: bits, characters, bytes, words, segments, blocks, tracks, tape reels, etc. These words refer to the physically accessible data elements that are recognizable by the hardware at various levels of accessibility.

Logical structure refers to the named data elements and their relationships. The named data elements are called items and can be categorized into compound (structured) items and simple (elementary and/or terminal) items. Compound items are items made up of other items; that is, they have a nested structure. Simple items are variables or names of properties or attributes of outside-world elements. They take data values, that is, they are unstructured fields in the data base which can be characterized as to type (binary, integer, alphanumeric, etc.) and length (fixed length or variable length). For example, MANUFACTURER may be the name of a simple item or field which, in a particular instance, in the data base has the value RCA. Synonomously, one may say the attribute MANUFACTURER has the value RCA. The outside-world elements that are described by the declarative data of the data base are messages, events, and/or objects which form the reason-for-being of the system and are the main concern of its users.

The term "data-base" itself can be thought of as the name of the maximum item of data — the root of a data structure tree and all its subnames. The data base is composed of lesser compound items called statements, files, records, and links. A file is a compound statement made up of an arbitrary number of statements called records which have the same logical structure. That is, a replicated item and its parent are called a record and a file, respectively. A link is an item which refers to another item which is physically remote, logically imbedded but physically non-imbedded. That is, a link is a reference to an item with more than one parent.

In some systems, the term "repeated group" or "repeated set" is used to refer to files that occur within records. The number of nesting levels of items within items is logically unlimited and may reflect the hierarchic structure of objects in the outside world (e.g., assemblies and sub-assemblies of components in equipment and systems).

In early tape-oriented data processing systems there tended to be a synonymy between logical and physical structure which led to unclear terminology and, unfortunately, some persistent confusion. For example, records and blocks, and files and tapes were not distinguished one from the other. Later, the term "physical record" was used for block and "physical file" for tape reel.

2.3.3 Directory Terminology

Although the data management system may have directories for other than data-base functions (e.g., program and job directories), the primary function of this paragraph is to discuss directories used for data access.

The scope of directories differs in different systems. A single directory (ordered listing) may cover the entire data base (as in RCDMS) or only a single file (as in IPS). A restricted directory scope may have the effect of requiring additional qualifying information from the user and a knowledge of the data structure, or files, in the system.

In addition to their scope, directories differ as to the type and depth of information derivable from them, and they are usually categorized in this way. Directories that are ordered by the names of items are called glossaries, term encoding tables, thesauri, term lists, etc. When restricted to terminal items, they may be called property lists, property rolls, attribute tables, etc.

Directories that supply physical format information are sometimes called format tables (e.g., IPS). When they supply information on logical structure, they are called logical indexes, item lists, or item position indexes. To point to specific records and fields, indexes are ordered by attribute and values.

Directory size and growth are dependent on both data structure complexity and data base size.

Term lists are primarily dependent on data structure complexity or vocabulary size, while indexes tend to have a base size which is dependent on data structure complexity and a variable component dependent on data base size. These relationships are summarized in Figure 2-1. When a data structure has been defined, the directory contains at least a term list, even with no data in the data base. This accounts for the intercept on the ordinate of Figure 2-1(b), which shows how the total directory size varies with the data base size for a fixed data structure complexity.

2.4 COMPARISON CHART

The aspects of data management systems listed in Paragraph 2.1 were evaluated for nine systems. A summary of the characteristics of these systems is given in the Comparison Chart (Figure 2-2) at the end of this section. In order to establish more clearly what system aspects were sought, a listing of these factors, under each chart heading, is given below:

2.4.1 User Languages

(1) Job Specification Language:

(a) Simplicity of job specification for the user.

(b) The ability to define multi-program jobs and their parameters.

(2) Query Language:

(a) Simplicity and format convenience for the user.

(b) Use of English vocabulary.

(c) The ability to integrate query output into other functions.

(d) The ability to specify search criteria of arbitrary logical complexity.

(e) The ability to specify format of output and display.

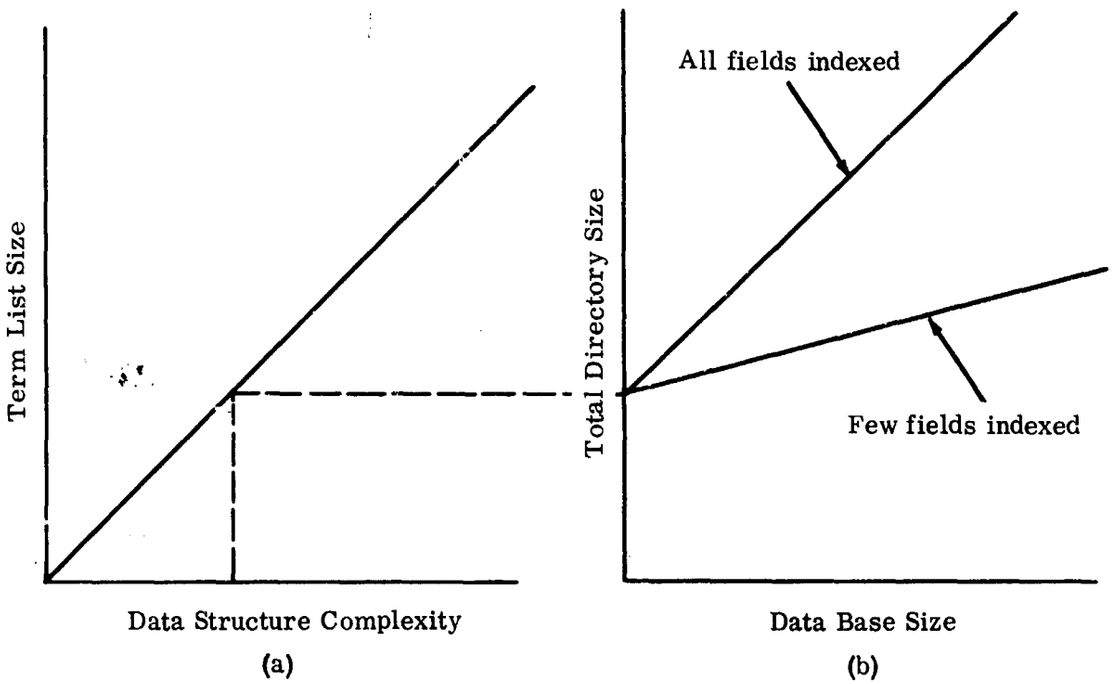


Figure 2-1. Directory Size and Growth

(3) Data Description Language:

(a) Ease of entry of new data terms, structure, and format into the system.

2.4.2 File Structure

(1) Directory:

(a) The existence of term dictionary, data structure, and format tables.

(b) The use of logical (structure defining) codes for data items.

(2) Data Indexing:

(a) The use of tables of addresses of data items by names and value.

(b) The ability to index selectively and thereby control index size.

(3) Data Structure:

(a) The ability to define fixed, variable, and optional items.

(b) The ability to define items which are imbedded (nested) in a parent item to an arbitrary depth.

(c) The ability to have a number of imbedded items at a given level in the structure of a parent item.

(d) The ability to cross reference, or link, to items having more than one parent item.

2.4.3 Processing Capability

(1) Search Logic:

(a) The use of logic operators AND, OR, and NOT in search criteria.

(b) The use of relational operators =, $\neq$, $\leq$, etc. as part of search condition terms.

(c) The use of functional and algebraic operators in search condition terms.

(2) Data/Directory Maintenance:

(a) System routines for generating and updating data and the directory.

(b) The ability to process data in on-line and batched modes.

(3) Output Formatting

(a) The existence of flexible output format generators.

(b) Sort routines.

(c) Generalized statistical routines.

(4) Job Management Services:

The existence of a system executive routine offering the following services:

(a) User job run control including program linkage.

(b) Program parameter binding.

(c) I/O services.

(d) Stored job descriptions (multiple program job descriptions)

(e) Priority scheduling.

(f) Expandable job repertoire.

2.4.4 Programmer Interface

(1) Procedural Language: The compilers or special languages provided.

(2) Program/Macro Library: The user-specific and service routines available.

(3) System Program Library Directory: Storage and maintenance of program parameter lists.

2.4.5 Response Time

(1) Storage Device: Tapes, Discs, etc.

(2) Search Strategy: The use of sequential versus random access techniques.

(3) Priority Processing: The existence of a quick-response mode.

(4) Levels of Accessibility: The use of more than one storage medium, and a file structure to take advantage of various media of different access times.

CHARACTERISTICS \ SYSTEMS		RCDMS	RECOL	ACSI-MATIC	ASO	IPS	
PROCESSING CAPABILITY	ANALYTICAL FUNCTIONS	Generalized and/or user-specific analytical programs process query output under control of job specification.	Arithmetic mean only.	None	Various inventory control routines.	Great circle distance; Latitude/longitude.	Great cir… Sums; Maximum…
	SEARCH LOGIC	AND, OR, NOT operators; =, ≠, >, <, ranges; Functional operators; Nested queries can search at any level in the data.	AND, OR, NOT operators; "Equal" relational operator only.	AND operator only, but other operators could be added.	AND operator; =, <, range relations.	AND, OR, NOT operators, =, ≠, <, > relations; Functional operators.	AND, OR… =, >, <… Functiona… No nestin…
	DATA/DIRECTORY MAINTENANCE	Off-line or on-line file generation. On-line data and directory updating; Generalized updating routines.	None; Both directory and data must be maintained off-line by special programs.	On-line updating of both data and directory; Generalized updating routines.	Automatic maintenance routines; Off-line file generation.	Automatic directory maintenance; Data update by user programs.	Data mai… by special…
	OUTPUT FORMATTING	Generalized report routine for printer and display; Eventually, query-directed format control.	Records may be merged, sorted and printed; Formatting is controlled by the query language.	Several different special reports were available; Generalized print routine.	Fixed report routines.	Generalized sort; Query-directed format control.	Query-di… format co…
	JOB MANAGEMENT SERVICES	Remote inquiry; Pre-stored job descriptions; Parameter binding; Job run control; Expandable job repertoire; I/O provided by environment.	None	Centralized I/O package; Dynamic allocation of core and peripheral storage; Minimum job specification.	Two priority levels, interrupt and save capability.	File maintenance and query tasks under control of IPS executive.	Remote in… I/O packa…
USER LANGUAGES	JOB SPECIFICATION LANGUAGE	An expandable, user-oriented job request language; Syntax driven scanner can process variety of languages.	None	A system language with a limited repertoire of job commands for specific system functions, and with a semi-free query format which uses tags to identify attribute values.	None	None (it runs in an operating system)	No separa… Jobs are …
	QUERY LANGUAGE	(Not developed) Executive routine provides for interface with query language.	Natural English-like query language; Arbitrary degree of nesting for logical conditions		Machine-oriented fixed format with coded fields.	English-like query language; Convenient format; Only one named file per query.	A simple, and Engli… untutored … language … Many files…
	DATA DESCRIPTION LANGUAGE	Data specialist-oriented data languages. User-oriented data formats.	None	None	None	None	None
FILE STRUCTURE	DIRECTORY	Term dictionary; Index from structure code to logical address; Format definition tables; Conversion to physical address.	Separate, simple directory for each file, giving format definition tables and field names (loaded by user at run time).	Term directory; Index from structure code to physical address; No format definition tables.	Hierarchic directory of attributes and values.	A separate term dictionary, format table, and physical index for each file.	Directory … location of…
	DATA INDEXING	Optional indexing by value to an arbitrary depth; Cross reference linkage in index.	None	Optional indexing by value to an arbitrary depth.	Linked list structure.	No indexing of field values.	None
	DATA STRUCTURE	Arbitrary nesting; Unlimited number of nested items; Fixed, variable and optional items; Cross referenced structure.	Fixed field and record lengths.	Data related hierarchically; Cross referencing; Frozen data structure, since no format definition tables.	Single-level unstructured data.	Flexible file structure; Unlimited nesting in depth and number of items; Variable length but no optional items.	Serial-par… with no ne… no cross r… between fi…
PROGRAMMER INTERFACE	PROCEDURAL LANGUAGE	JOVIAL; CS-1; TRIM.	Not applicable	A special ACSI-MATIC programming language with attendant assembler.	None	Special problem-oriented language for file maintenance routines.	No proced… Programs … assembly … called "AP…
	PROGRAM/MACRO LIBRARY	Automatic maintenance of library; Expandable program repertoire.	None	Macro library controlled by assembly program.	System controlled program library.	System controlled macro library, and library of update routines.	Program s… under cont… executive.
	SYSTEM PROGRAM LIBRARY DIRECTORY	Complete program directory containing sufficient parametric data to permit automatic linking of programs as specified by job description.	None	None	System maintained.	No program directory; Library tape is scanned to find segment associated with file being processed.	Automatic … programs … Frequently … be saved.
RESPONSE TIME	STORAGE DEVICE	Auxiliary core; High-speed drum; Disc; Disc-simulated magnetic tape.	Not more than one magnetic tape reel for each file.	Disc; Disc-simulated magnetic tape.	Disc	Multi-reel magnetic tape.	Disc.
	SEARCH STRATEGY	Optimized, directory-driven random access search strategy using inverted list structures	Linear search through a serial file using a single, generalized search routine.	Directory-driven random access search strategy using a list-type structure.	Combined index and serial search.	Linear search through a serial file using a single, generalized search routine.	Linear sea… parallel da…
	PRIORITY PROCESSING	Priority job processing controlled by facility management and users	Queries may be batched.	None	Two-level priority.	None (except that provided by operating system environment).	Priority pr… program c… levels.
	LEVELS OF ACCESSIBILITY	Multi-level accessibility on disc; Tape back-up; Fast access system drum and system core.	Single level on magnetic tape.	Two levels on disc and magnetic tape.	Single level on disc.	Single level on tape.	Single level…

ECOL	ACSI-MATIC	ASO	IPS	473L	COLINGO	LUCID	ADAM
ean only.	None	Various inventory control routines.	Great circle distance; Latitude/longitude.	Great circle distance; Sums; Maximum/minimum values.	Mathematical and logical operations under control of the query language.	Sum and arithmetic mean, as part of query.	Analytical functions can be accommodated when defined as part of a system being evaluated by ADAM.
OT operators; ional operator	AND operator only, but other operators could be added.	AND operator; =, ≠, <, range relations.	AND, OR, NOT operators; =, ≠, <, > relations; Functional operators.	AND, OR operators; =, >, < relations; Functional operators; No nesting.	AND, OR operators; =, >, < relations; Search at any level; Functional operators require two passes of file.	AND, OR, NOT operators; =, ≠, >, <, relations.	Handles general search criteria as specified by user.
rectory and data tained off-line by ams.	On-line updating of both data and directory; Generalized updating routines.	Automatic maintenance routines; Off-line file generation.	Automatic directory maintenance; Data update by user programs.	Data maintenance performed by special service programs.	Automatic system maintenance routines for both directory and data.	Changes to data base description require a complete recompilation.	Off-line data generation; Automatic directory maintenance of structure (name) changes.
be merged, sorted controlled by the	Several different special reports were available; Generalized print routine.	Fixed report routines.	Generalized sort; Query-directed format control.	Query-directed sort and format control.	Report generators; Sort routine.	Generalized output facilities for console display and printing.	Output control provided; Output routines are user provided.
	Centralized I/O package; Dynamic allocation of core and peripheral storage; Minimum job specification.	Two priority levels, interrupt and save capability.	File maintenance and query tasks under control of IPS executive.	Remote inquiry; I/O package.	Executive routine executes control language messages.	No current provision for executing independent user analytical programs against data base.	Job run control provided; Entire job must exist in core. Remote inquiry.
	A system language with a limited repertoire of job commands for specific system functions, and with a semi-free query format which uses tags to identify attribute values.	None	None (it runs in an operating system).	No separate job language; Jobs are initiated by query.	User job and query functions defined and initiated with a control language; A two-level query structure for simple or complex queries.	None	ADAM translator accepts user-defined language.
h-like query ee of nesting for ns		Machine-oriented fixed format with coded fields.	English-like query language; Convenient format; Only one named file per query.	A simple, uniform punctuation and English-like syntax for untutored users; A more powerful language for complex queries; Many files per query.		An English-like language with a convenient user format.	Automatic translation of user-defined query language. Execution of query follows
	None	None	None	None	COBOL-type data description language.	User-oriented data formats.	ADAM translator accepts user-defined language.
ble directory for g format e and field names r at run time).	Term directory; Index from structure code to physical address; No format definition tables.	Hierarchic directory of attributes and values.	A separate term dictionary, format table, and physical index for each file.	Directory to the name and location of each file.	File directory giving name, location, and format of system files.	Concordance dictionary with complete cross reference of data values.	Term dictionary; Logical name to physical location indexing; Format definition tables.
	Optional indexing by value to an arbitrary depth.	Linked list structure.	No indexing of field values.	None	None	Complete indexing by data values.	To first level by name only.
record	Data related hierarchically; Cross referencing; Frozen data structure, since no format definition tables.	Single-level unstructured data.	Flexible file structure; Unlimited nesting in depth and number of items; Variable length but no optional items.	Serial-parallel data storage, with no nesting of files and no cross referencing of data between files.	Two-level data structure; One nested item with up to 14 records.	No implied structure to data base; No embedded structure in concordance; Subsequent organization into two levels.	Arbitrary depth of nesting; Unlimited number of nested items (repeat groups); Fixed and variable items.
	A special ACSI-MATIC programming language with attendant assembler.	None	Special problem-oriented language for file maintenance routines.	No procedural language; Programs coded in a basic assembly language called LAP.	COBOL.	JOVIAL; FAP.	FORTRAN; DAMSEL.
	Macro library controlled by assembly program.	System controlled program library.	System controlled macro library, and library of update routines.	Program and macro library under control of the system executive.	Program/macro library maintained by system.	System programs in library with no automatic maintenance.	Macro library; Service routines.
	None	System maintained.	No program directory; Library tape is scanned to find segment associated with file being processed.	Automatic assignment of programs to satisfy queries; Frequently used queries can be saved.	System maintained.	None in pilot version.	Compiler controlled.
ne magnetic ch file.	Disc; Disc-simulated magnetic tape.	Disc	Multi-reel magnetic tape.	Disc.	Programs on disc; Data files on magnetic tape; Some data on disc.	High-speed drum; Tapes.	Disc; Magnetic tape.
through a serial gle, generalized	Directory-driven random access search strategy using a list-type structure.	Combined index and serial search.	Linear search through a serial file using a single, generalized search routine.	Linear search of serial-parallel data or random access.	Linear search through a serial file using a single, generalized search routine.	Random access search strategy using concordance; No linear search required.	Random access to named item; Sequential search for specific criteria.
batched.	None	Two-level priority.	None (except that provided by operating system environment).	Priority processing by program category, three levels.	None	None in pilot version.	Priority job processing controlled by facility management and users.
magnetic	Two levels on disc and magnetic tape.	Single level on disc.	Single level on tape.	Single level on disc.	Two levels, Disc for programs and high priority data, Tapes for most data.	Directories on drum; Data on tapes.	User defined.

Figure 2-2. Comparison Chart

SECTION III. DESCRIPTION OF THE RELIABILITY CENTRAL DATA MANAGEMENT SUBSYSTEM

3.1 OBJECTIVES OF THE RELIABILITY CENTRAL

The Reliability Central Data Management Subsystem (RCDMS) is that part of the Reliability Central which handles all reliability data to be processed by electronic means. To understand the objectives of RCDMS, it is important first to understand the objectives of the Reliability Central as a whole. These objectives are listed below in terms of the functions assigned to the Reliability Central:

(1) Mission — The Reliability Central will serve the Air Force as the central management activity for the control of reliability information, with the objective of improving the reliability of Air Force equipment.

(2) Data Management — The Reliability Central will provide a repository of reliability information which will include validated manufacturers' reliability data, acceptance test data and field evaluations of parts used, and expected to be used, in Air Force systems. The responsibility of managing the repository will include acquisition, validation, reduction, correlation, analysis, storage, and dissemination of the reliability data.

(3) Analysis — The repository of the Reliability Central will not only contain raw test data describing parts reliability, but it will also contain summaries and analyses of the data stored in the forms most convenient for the users of the Reliability Central. The Reliability Central will analyze

data at the time of conversion from raw form to summary form, and it will also perform analyses on a demand basis in conjunction with queries initiated by users of the system. The system will be designed so that a single analysis can be made from data which resides in several different files.

(4) Dissemination — The Reliability Central will disseminate the data on both an automatic and demand basis to System Project Officers, contractors, subcontractors, and other authorized users. Automatic dissemination will be in the form of predefined reports issued periodically on all aspects of reliability qualification, part design, selection, test, and application. Demand dissemination will be in the form of specific responses to queries which can have an arbitrary logical complexity.

(5) Maintenance — The Reliability Central will provide for the maintenance of the data base in a fashion sufficiently flexible and dynamic to permit the creation of new file structures from old data with minimum cost and delay. There will be many kinds of files, each with its own structure. The ultimate goal is that the system will be responsive to changes in usage habits so that the system can be self-adjusting. The adjustment would consist of an automatic restructuring of any file to make it better suited for anticipated usage in the future. There are no constraints on the format, structure, or content of new information to be added

to the data base. New information must be able to be added by creating new files or records in the data base or by incorporating new information in existing data structures.

(6) Query Access — The Reliability Central will stress the importance of giving the user convenient access to the data in two ways. First, the user will be permitted to state his query in a manner which is simple and which requires no knowledge about the operation of the system. Second, the user will be given the choice of having almost immediate access to the data or of having his query processed on a scheduled basis. Ultimately, the design will permit the user to state in his query the analytical functions to be carried out on the data retrieved in response to his query.

3.2 CHARACTERISTICS OF THE DATA BASE

A description of the data base to be managed by the Reliability Central Data Management Subsystem will give the reader a better understanding of the objectives and functions of RCDMS. The specific structure of the data base cannot be described because it will be evolutionary and will depend on the needs of the users. The general nature can be described, however, and is set forth below in terms of size and structure. Detailed treatment of the file structure design is available in AUERBACH Technical Report, 1193-TR-2, produced under AF contract AF30(602)3433.

3.2.1 Data Base Size

The design places no restriction on the ultimate size of the data base. The Reliability Central Data Management Subsystem is designed to accommodate an interim operation called the Test Operation. The Test Operation will include all the basic system functions and a large part of the ultimate data base.

The size of the data base for the Test Operation is estimated to be up to 440 million characters, of which 400 million characters are expected to be raw data and 40 million characters are expected to be summary data. In addition, there will be an estimated need for at least 20 percent of the data base to store the directories which control the data base. If an extensive depth of individual field indexing is called for, the storage requirement for directories could expand substantially.

3.2.2 Data Base Structure

The data in the data base will require varying degrees of accessibility. The raw reliability data resulting from manufacturers' tests, acceptance tests, and field tests will be accessed relatively infrequently. After the data has been analyzed and reduced, it will be retained in test summary form and will be accessed more frequently. The most frequently used data will be placed in component summaries. It is anticipated that the component summaries will be arranged by major component type (e.g., transistor), generic class code or IDEP number (e.g., silicon, NPN, power), manufacturer's process family (e.g., Fairchild Type 100), and part number (e.g., 2N1605). This general arrangement will have a number of variations depending on the type of data being retained and on the use to which it is being put.

3.3 DESIGN OBJECTIVES OF THE DATA MANAGEMENT SUBSYSTEM

The general objective of RCDMS is to provide the Reliability Central with the data management capabilities outlined below. Since the Reliability Central will be an operational rather than a research facility, it is imperative that the objectives of RCDMS remain pragmatic rather than take on a speculative tone. In order to meet overall Reliability Central objectives, the RCDMS designers have had to incorporate many general-purpose data management concepts. As a result, when the design is implemented, it will not only serve the Reliability Central effectively and efficiently but it should be equally applicable to a variety of other problems which require management of a large data base.

RCDMS is designed to meet the following specific objectives:

3.3.1 Convenience to the User

The overriding theme of RCDMS is that it be a service tool to a multitude of different users who must be accommodated quickly, effectively, and inexpensively. This theme permeates all of the design objectives below.

It is assumed that the types of users will range from those who know nothing about the system (and who wish to use it without learning more) to those who understand the system well (and who wish to manipulate its inner workings to their advantage). Consequently, the system will have a convenient, user-oriented language which shields some users from the complexities of the system. On the other hand, the system will be sufficiently flexible (and well documented) to permit the knowledgeable user to exploit facilities of the system in order to achieve more than the usual results.

3.3.2 Capability

A subsystem design must be capable of meeting the objectives of the system as a whole. The RCDMS objectives for a capability which meets the Reliability Central objectives are discussed below:

(1) Job Specification — The system will have the capability of storing both generalized system programs and specialized user programs so that they can be called upon by a user to be run in the sequence he desires. This job specification capability should be effected by means of a command language which is both easy to use and comprehensive in its ability to specify jobs. Each job will be checked out to the fullest extent possible at the time of storing so that the system can assure the maximum probability that a job will run successfully when it is called.

(2) File Structure — The system does not specify or imply a data structure constrained by the nature of the system's design, but will provide a framework for building any reasonable logical data structure desired by the user. A data description language will be provided which is easy to use and which permits specification of any data to be included in the data base in a logical and consistent manner. The system design will provide for a wide variety of files, each with its own data structure. The records of the files may contain such varied data items as variable length fields, embedded files, linked files, optional fields, and the like. The system will also be able to convert a structure from one form to another, at the direction of the data base management staff. Regardless of the file structure in existence at any one time, the totality of the data base is accessible to the user for retrieval of any elements specified in a query statement.

(3) Query Language — RCDMS is being designed to incorporate user-oriented query languages or other user-specified languages. It achieves this capability by utilizing an input interpreter which is driven by a syntax table. RCDMS provides the basic capability of handling conditional searches of arbitrary logical complexity, in anticipation of the query language. RCDMS design provides for an interface with the anticipated query language through a flexible interface among the query language, the search strategy system, and the RCDMS supervisor.

(4) Compatibility With Environment — RCDMS will be operating at least during the Test Operation on the Rome Air Development Center computer complex in Rome, New York. It is designed to interface with, and take full advantage of, the RADC Executive Control System.

3.3.3 Adaptability

All systems exist in an environment of change. The systems that adapt are those that continue to give useful service. Rigid systems soon outlive their usefulness, and are replaced. Several adaptive aspects of RCDMS are discussed below:

(1) Modular Design — The design of RCDMS is modular in concept so that it can be expanded with minimum cost and effort. The initial implementation of the design will be for the Test Operation, which is an interim operation involving the basic operation of the system without full capacity or full capability. At a later time, the size of the data base will be increased and the capability of the system will be extended. The design provides for incremental growth of the data base and the capability and functions of the system, without the need to redesign and reprogram the initially implemented basic elements of the system.

(2) Restructuring of Data — It is anticipated that the needs of the reliability specialists and contractors using the system will change and expand, thereby requiring dynamic changes in the structure of the data base in order to maintain operational

effectiveness. In response to a user command, RCDMS will be able to extract data from its existing logical structure and rearrange it into a structure more suitable to current use. Furthermore, the system will retain statistics on data usage so that it will be able to respond automatically to significant changes in usage patterns, and will be able to assist the data management personnel in changing data structure. It is conceivable that ultimately the system will be able to analyze the new usage need, formulate its own request for the change in data structure, and execute the change automatically, however this is not an initial design objective.

(3) General Purpose — The primary responsibility of the RCDMS is to manage the data base and processing functions of the Reliability Central. However, the RCDMS design will be able to manage many kinds of data bases and functions, following its initial application in the Reliability Central. As previously noted, no truly general-purpose data management system has hitherto been implemented for production operation, even though the need for data management is pressing in most branches of the Government. The Reliability Central hopes to make a significant contribution beyond the present state of the art in this area.

3.3.4 Cost

The reduction of cost is a major objective in all system design. Since the Reliability Central is an operational rather than a research facility, cost takes on added importance. There are two large areas of potential cost reduction in RCDMS:

(1) Implementation Cost — The RCDMS design will consist of a large number of generalized task subroutines. Many subroutines will be suitable for use in several parts of the system under varying circumstances. This multiple utilization of subroutines will tend to result in a reduction in programming cost during the implementation stage. A further cost reduction should result from the modular design. When the system is expanded from the Test Operation capability to the ultimate system capability, the modular design should provide a stable foundation to which new functions can be added with minimum cost.

(2) Operating Cost — By making RCDMS job-oriented (as described in Paragraph 3.3.2 (1) above), the cost of operating the system should be reduced significantly because new jobs can be initiated with relatively little new programming.

If all of the system functions are performed by generalized programs that can be linked together in many combinations the user is required to program only the special functions associated with his particular need. RCDMS would serve these special functions in a supporting role with no need for costly reprogramming of the system.

3.3.5 Response Time

In line with the objective of giving maximum convenience to the user, RCDMS is designed to communicate its results back to the user at the earliest possible moment, consistent with reasonable cost. The design objective of RCDMS is to provide the fastest response time possible with the state of the art by means of console access. Many questions and jobs directed toward the data base will require a rapid response for the information to be entirely useful. In addition, experience has shown that queries become more effective if the user is provided with immediate feedback. The user has the opportunity to improve the query by modifying it. Where scheduled outputs are called for which require stepping through the data base in an orderly manner, the system also provides facilities for stepping through the data using techniques which minimize directory search.

To achieve the system speed and balance commensurate with the concept of console access, RCDMS is designed to meet the following objectives:

(1) Random Access — Effective console access demands random access to the data base. The advantages of console access can be quickly vitiated if the system must take the time to perform a linear search through a serial data store, such as a magnetic tape, while the user waits at the console.

(2) Data Indexing — Random access storage, in turn, demands a powerful data indexing scheme in order to reap the full processing potential from the storage medium. If a restrictive data indexing scheme is employed, an inappropriate portion of time is spent to access data, thereby increasing response time and cost. RCDMS

employs a data indexing scheme which permits the system to retrieve from storage only those data items that meet the exact criteria of the query or search regardless of the distribution of the indexed data sought among the files comprising the data base.

(3) Levels of Accessibility — A corollary to the problem of logical access optimization is the problem of physical accessibility of the data. A large computer facility, such as the RADC complex, normally employs several levels of storage media — each with its own access speed. To reduce overall response time, it is important to allocate frequently accessed data to the storage medium with the fastest access time. RCDMS will record the access frequency of various data groups, and transmit this information to the RADC complex executive system in a manner compatible with the complex, to guide the physical location of data.

(4) Priority Processing — Some queries or jobs are more important than others. A set of priorities will be established by data management personnel. Ultimately, RCDMS will be designed to recognize the priorities and to stack the queries or jobs according to their priority. The RCDMS design is consistent with a processing capability which organizes requests for parallel processing of queries with like priority, and can possess this capability itself, if the system is modified.

3.4 CHARACTERISTICS OF THE DATA MANAGEMENT SUBSYSTEM

The Reliability Central Data Management Subsystem has been designed with sufficient flexibility to handle many types of data in addition to reliability data. The principal objective of the system, however, is to provide a framework within which reliability data may be analyzed, evaluated, summarized, and stored in such a way that it may be retrieved easily and quickly by reliability specialists. Two of the system elements required to perform this function are the common data base and the five user job request types with their attendant programs. The other system elements are the directories which point to the data, the system programs which use the directories to manipulate the data, and the Supervisor program which controls the entire Reliability Central Data Management Subsystem.

This section summarizes the functions of the five system elements in order to illustrate how they work together. More detailed information on the subject can be found in Volume I of this Final Report, specifically in Sections V and VI.

3.4.1 Common Data Base

Since reliability data is the basic element, or resource, of the Reliability Central Data Management Subsystem, the fundamental strategy of the system is to retain the data in as flexible and accessible a form as possible. The RCDMS data description language not only permits the use of variable length fields and optional items, but it also permits nested structures such as the nesting of a variable length file within a record of another file. Another feature is the ability to logically link to items that are stored physically with other generically related items, so that a given item may be part of more than one parent item, in the sense of a lattice-type hierarchy. These links are kept in the RCDMS directory so that the identity of all items to be retrieved is known before any data access is made, and no access need be made to the parent data which originates the link.

The data itself will be either in random access or in magnetic tape storage under the control of the Local Control Program (LCP) of the RCDMS computer and Executive Control Program (ECP) of the RADC EDP Complex. A preference for one medium or the other may be stated by RCDMS. Since the ECP can change the physical location of the data without having to notify RCDMS, data will be requested by means of a logical name which will not change in spite of any physical data movement.

The logical name of a data item is derived from the relative position of the item within the structure of the data base. A unique code may be created for each item in the data base. The logical code is a numeric representation of the nodes in the multi-list tree structure of the data base, and is called the Item Position Code. Figure 3-1 shows a hypothetical part of the Reliability Central data base. The data items are indicated as branches off a central stem. Each branch is numbered, in relation to other branches. A file contains records, or repeated items, which can represent an indefinite number of branches. Each node has its own code.

RCDMS supplies a standard language for defining the logical structure of data in the common data base. The language is supplied in two versions. The first is a simplified version which uses an indented outline type structure on a standard form to signify the relationships between data items and subitems. The second is a more complex version which uses a formal parenthetic punctuation to signify data class relationships. This version is intended for those users who have more knowledge of the system, and who wish to use a linear parenthetic string input rather than a columnar page format. (The system may make a translation from the simple to the complex version internally before using it for processing.)

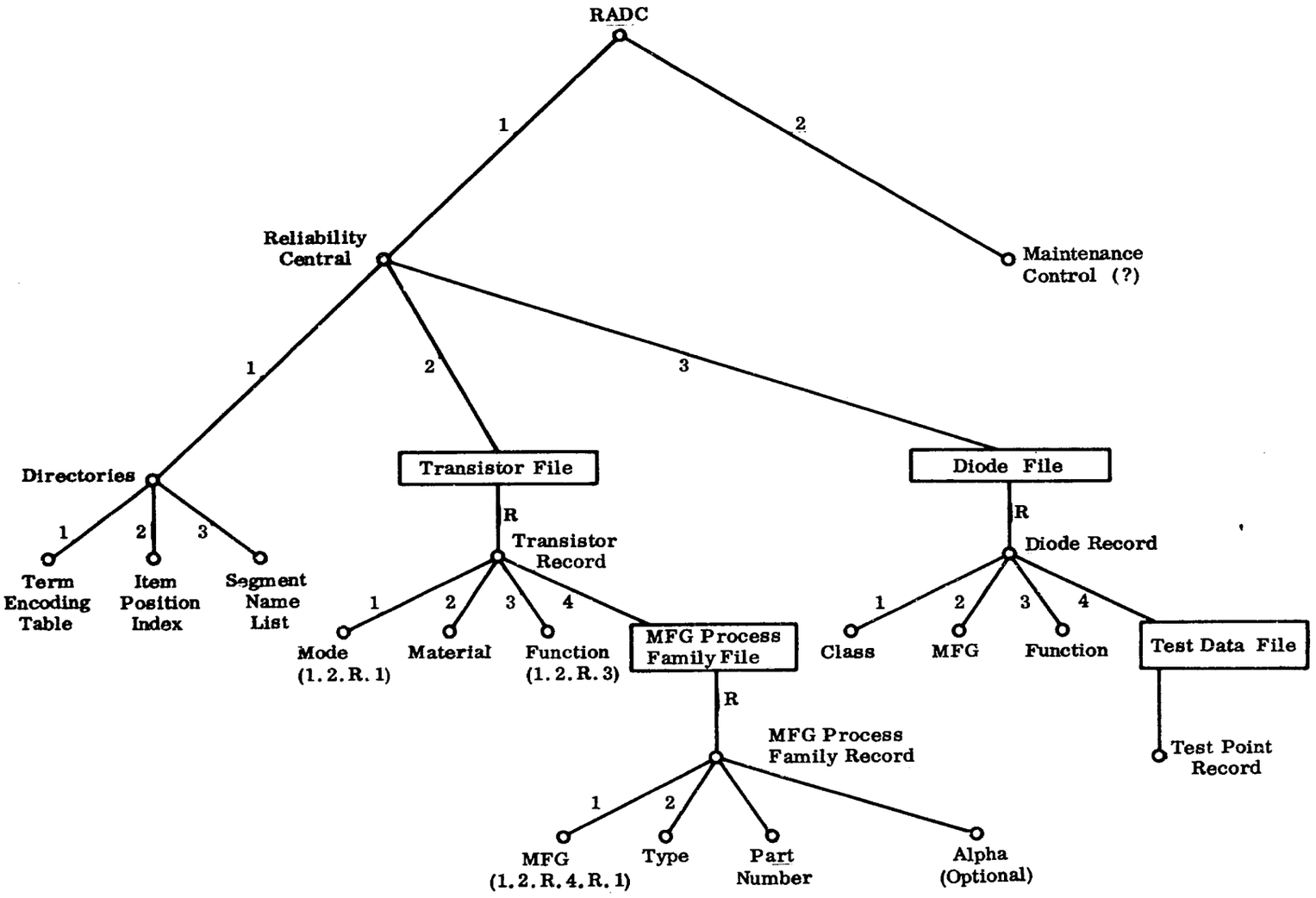


Figure 3-1. Tree Structure Example

In addition to the standard language, RCDMS has the capability of translating and understanding a data description (or other command) language which the user himself has created. The user can specify the syntactical and semantic rules which are appropriate for his mission. He can inform the system of these rules by means of action graphs (which must be compiled with the system). The system uses the action graphs to control its scan of the user's input, and the system actions specified by the user are taken at the appropriate time.

3.4.2 Directories

RCDMS provides the user with a definitive data description language which permits him to specify a very wide range of data structures. The structural description of the data is implemented by means of the RCDMS directories, which tie the system to the data base. By giving the user a flexible language for defining the data logically, isolated from the problems of handling the data physically, RCDMS achieves a new level of generality.

The reliability data can be recalled to core memory only by means of system directories. A function of the directories is to translate the names of data items, first, into logical codes which describe the relative or logical positions of the items, and, second, into the name of the segment which can be used by the RADC Local Control Program (LCP) to fetch the data. A segment is a block of 1024 M1218 words, equivalent to three blocks of 512, 12-bit words, used as the RADC data transfer standard. All system functions of file maintenance and retrieval depend on the directories to locate the desired data and to describe it once it has been found. There are four main directories to the system. Most of the directories will be quite large and will themselves be kept in random access storage, thereby requiring their own small subsidiary directory in core to permit fast access to the main directories by the system.

Figure 3-2 shows what each directory requires as input, and what each is designed to provide. There are several functions of the data directories:

(1) They are utilized to focus in on the data;

(2) They are utilized to give a description of the data in the record (e.g., whether the data is a floating point or an alphanumeric number);

(3) They are utilized to permit extraction of the data;

(4) They contain the index values for the data items so that searches can be performed within the directories without having to access the data until the end of the search, when only the correct data items are drawn from storage.

The description given by the directories is general in nature so that many types of file structures are permitted and the programs that operate upon the directories do so in a general manner.

(1) Term Encoding Table — The basic function of the Term Encoding Table is to convert the name of a data item from its alphanumeric input form to a coded form which describes the logical position of the item in the data structure. The coded form, called an Item Class Code (ICC), consists of integers which represent the nodes on the tree structure describing the data. For example, the ICC for the Transistor File in Figure 3-1 would be 1.2. The code may be derived in reverse by tracing the nodes back through the tree structure to the root. Thus, the Transistor File is the second node of the first node, or 1.2.

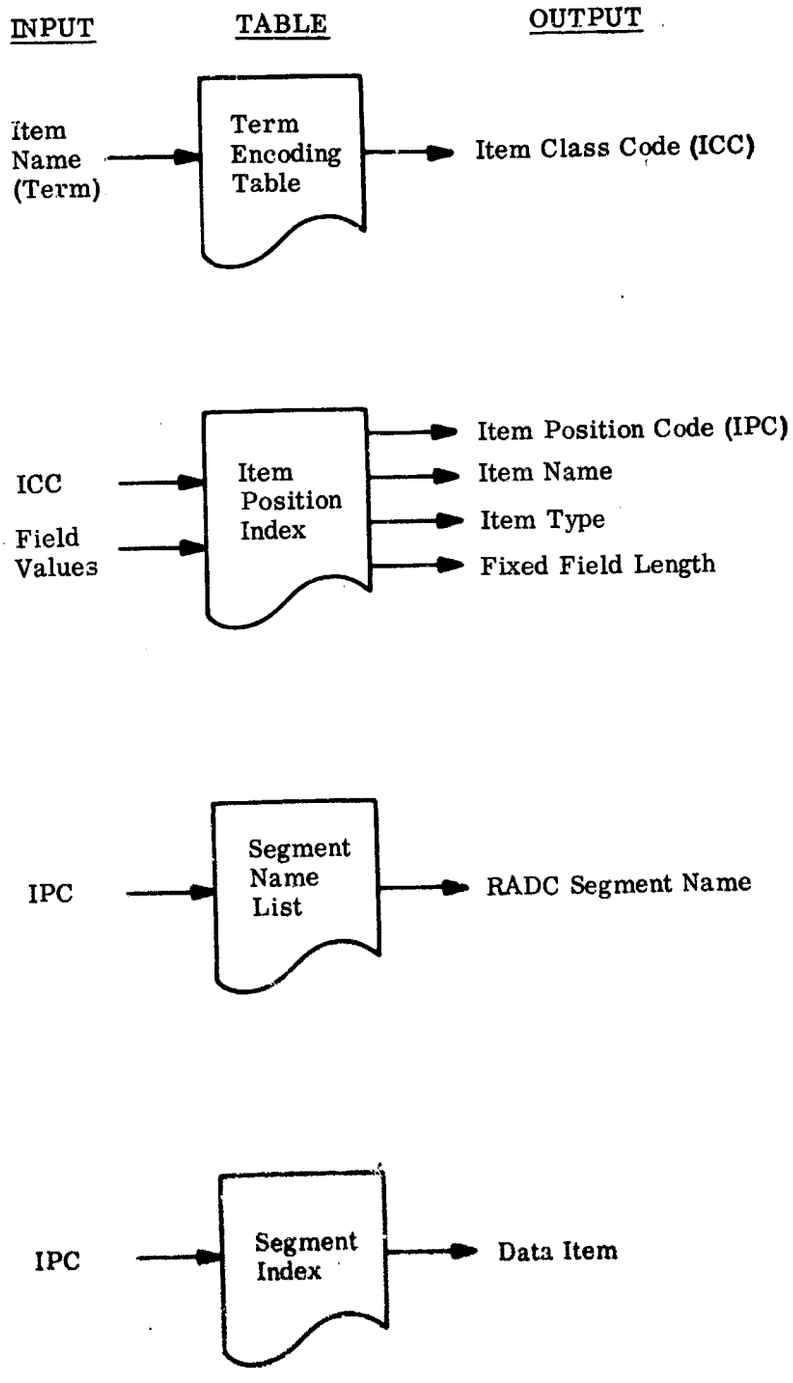


Figure 3-2. Directory Inputs and Outputs

When a node is a file, it may contain any number of records, each of which is a subordinate node. This class of nodes is represented by the letter R in the ICCs for items within a record. For instance, the field called TYPE in Figure 3-1 would have an ICC of 1.2.R.1. When a series of files are nested within the records of other files, several R's can be contained in one ICC, such as the fields of Test Point Record in Figure 3-1. The ICC in this case is not unique to any one item, but it describes a class of items within the files; hence the name: Item Class Code (ICC).

A given term will have more than one ICC associated with it if it is used more than once in the data structure. This would be the case if TYPE were the name of a field in both the Transistor File and the Diode File. Part of the program function will be to resolve such ambiguities, using qualifying context or names.

(2) Item Position Index — The Item Class Code must be converted into a unique code by supplying values for the R's before a specific item can be retrieved from random access storage. The conversion is performed by the Item Position Index, which is arranged by ICC and contains all of the values by which certain data items have been indexed. When the ICC has been changed into a unique set of integers with no R values, it is called an Item Position Code (IPC).

The Item Position Index has two alternate ways of creating an IPC. First, it can allow a program to generate a series of successive R values for an ICC, which creates a series of IPCs and permits the system to step through a file one record at a time. Second, it can provide the one R value for the unique IPC which points to the record satisfying some search criteria (by means of field value and R-value index tables).

The Item Position Index supplies other information descriptive of the data, such as the Item Type. The item may be specified as either a required or optional statement, file, record, or field. If the item is a field, it is specified either as a fixed or variable length field. The length of fixed fields is also specified. The Item Type also describes whether the data is floating or fixed point, alphanumeric, or other mode.

A vital function of the Item Position Index is to retain indexing data. This data corresponds to the addresses of data items which have certain indexed values. Since logical addresses are being used instead of physical addresses, the RCDMS design can retain the address in a very small space, often as little as one digit. The power of these indexes comes into play when complex searches for data items can be accomplished without making any unnecessary accesses to peripheral storage. The Item Position index can retain index data to any desired level.

Frequently accessed data can be more fully indexed, for instance, than data which is needed less frequently. This flexibility permits the personnel operating the Reliability Central to specify the level of indexing which will produce the minimum expense in the trade-off between storage cost for the index data and accessing cost for retrieving unindexed data.

Another function of the Item Position Index is to retain the statistical tallies of data usage since all usage must pass through the Item Position Index. It is from these tallies that the need for data restructuring will become evident. The restructuring will be done by reliability specialists in the beginning. Ultimately it is hoped that the system will perform its own restructuring. The tallies are also used to communicate the frequency of usage to the data base manager of the RADC complex to indicate where the data should be stored physically in terms of accessibility.

When the Item Class Code has been converted into a specific Item Position Code, it is ready for further conversion into the name of a specific segment of data which contains the item.

(3) Segment Name List — An Item Position Code is a code for a unique data item. It embodies sufficient information to be able to call for its appropriate data segment, which is a section of the data base with 1024 18-bit words. Determining the name of the segment to be called is the task of the Segment Name List. When the name has been determined, it is given to the Local Control Program (LCP) along with a request for the segment, and the LCP retrieves the segment from peripheral storage placing it in a prescribed input area in core memory where the exact data item can be located.

(4) Segment Index — The final step in retrieving a data item is taken with the aid of the Segment Index. Each segment of data begins with a Segment Index which points to the data items contained in the segment. The Segment Index uses the IPC and auxiliary information from the Item Position Index to find its way to the desired item. At this point, there are several things which may be done with the data. The item may simply be retrieved, it may be deleted from the segment, or it may be used as a foundation upon which to add new data items to the file.

3.4.3 Job Specification

A job is defined as a sequence of program tasks that accomplishes some desired user action. The Reliability Central Data Management Subsystem is designed

to perform a variety of different jobs with a minimum of programming through the use of many generalized programs. The user will be able to call on these programs in the sequence he desires. It is this aspect which gives RCDMS much of its general purpose nature. The user accomplishes this variety of uses by means of a Job Request. The Job Requests perform the functions of item definition, data entry, program entry, job entry, and job running. The first four Job Request types perform system maintenance functions. The fifth Job Request type permits the user to run jobs of his own on the system. A standard command language is provided for the user to specify each job he wishes accomplished. An additional feature of RCDMS is that the user can also request creation of his own command language, using syntactical and semantic rules which are more closely associated with his mission than is the standard language.

Job specification affords both a dynamic and modular expansion to RCDMS. As to the data base, new data can be added in batches or on-line, with few constraints on the type of data. As to the system functions, modular additions can be made which incorporate new task-oriented programs. The Job Request types are summarized below:

(1) Item Definition Request – The logical structure of the data base can be altered through an Item Definition Request. An alteration is made in the directory entry which describes the relationship of the data item to the data base. Possible alterations include the addition or deletion of data items such as statements files, records, or fields.

(2) Data Entry Request – Data may be added or deleted by issuing a Data Entry Request. This type of request would normally be used for transactions where the data in the data

base does not have to be examined in order to complete the transaction. More complex transactions may be effected by combining the Job Entry Request and Job Run Request described below. Data may be entered by means of a console or by magnetic tape. The data may be in the internal format of the system, or it may be in the external format of the data description language.

(3) Program Entry Request — Descriptions of programs are entered into the system with the Program Entry Request. The programs and the descriptions of their parameters are given a unique name and are placed in the Program Description List, where they are on call for the execution of jobs at a later time. Program Descriptions are used for checking purposes to be certain that subsequent jobs, which call on the programs do so correctly and with the proper parameter specifications.

(4) Job Entry Request — The Job Entry Request is used to bring a Job Description into the system. A Job Description specifies the programs to be run for a job, and the sequence for running the programs. Any predetermined program parameters for the job may also be entered by this means. A Job Description is usually entered for a job which is run frequently, so that the job does not have to be specified each

time it is run. A Job Entry Request causes a Job Description to be stored so that it may be called later by a Job Run Request. Job Descriptions are checked carefully by the system upon entry to be certain that they are compatible with the requirements of the programs they use.

(5) Job Run Request — A Job Run Request asks for a particular job to be run. The job must have been described previously in a Job Description by means of a Job Entry Request. If the Job Description specifies all parameters, the Job Run Request can be as brief as to simply identify the job name to be run. However, if the Job Description refers to several programs requiring parameters which have not been pre-assigned in the Job Description, the Job Run Request must specify these parameters in detail.

3.4.4 Supervisor Program

The Reliability Central Data Processing Subsystem derives its primary control from the Supervisor Program. A graphic representation of the Supervisor Program may be seen in Figure 3-3. The Supervisor Program is subservient to a Local Control Program (LCP) which in turn is subservient to the RADC Executive Control Program (ECP). The ECP supplies all inputs and receives all outputs through the Local Control Program (LCP).

(1) Job Request Processor — The portion of the Supervisor Program which receives all Job Requests from the user is

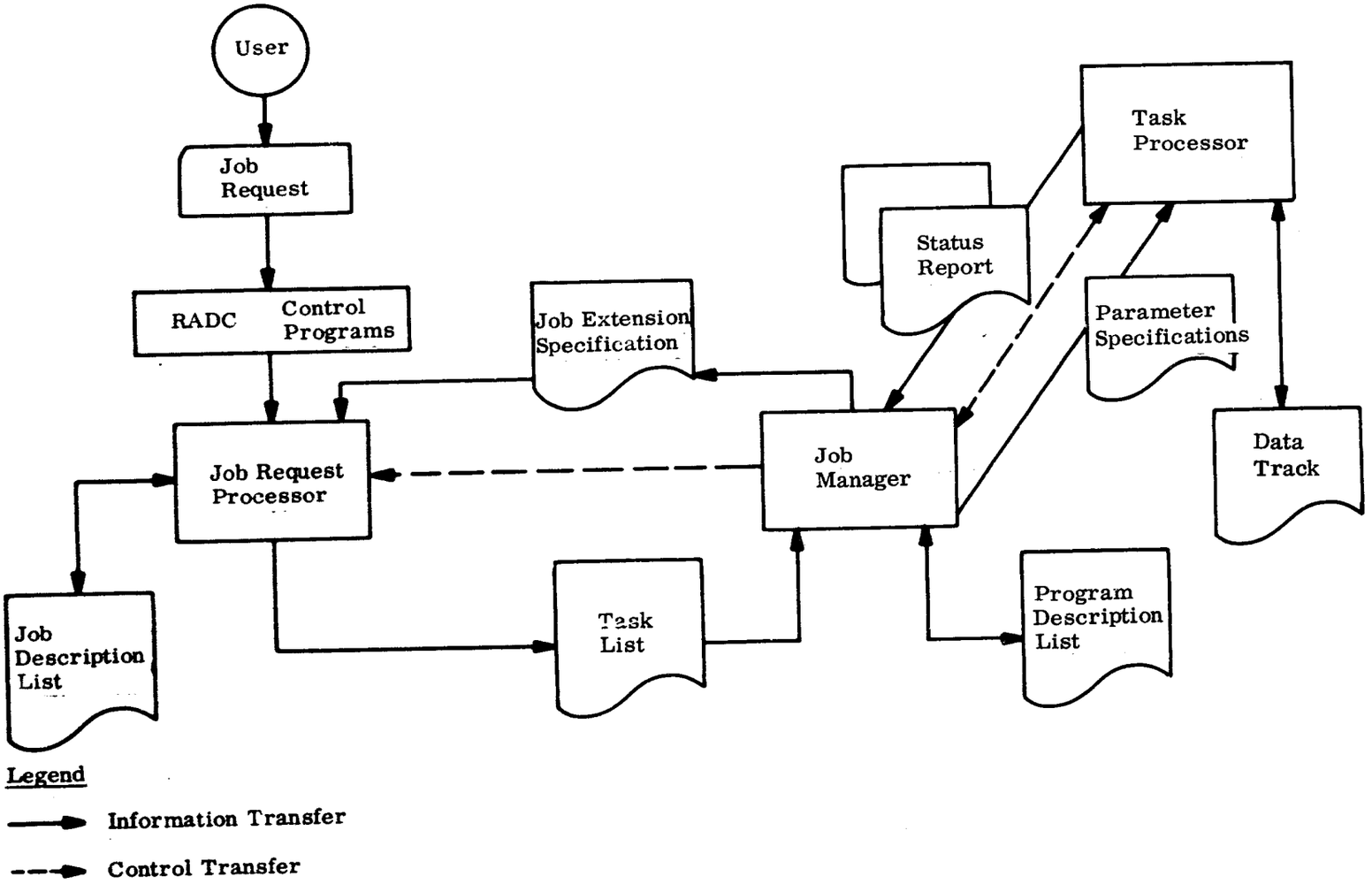


Figure 3-3. Supervisor Program

called the Job Request Processor. The first function of the Job Request Processor is to interpret the Job Request and determine which job is to be done. The Input Scanner, a subroutine in the Job Request Processor, plays a large part in this by analyzing the syntax and meaning of the Job Request. The Input Scanner is driven by a table called an Action Graph which specifies the syntax of the input language and the actions to be taken by the Input Scanner. New user languages can be introduced into the system by preparing new action graphs for the Input Scanner. This capability permits system specialists to make evolutionary changes in the user language without having to reprogram the Input Scanner.

The Job Description which matches the Job Request is fetched from the Job Description List. The Job Description is comprised of the program names required to do the job, along with the sequence of the programs and any prescribed program parameters. The parameter values which have been presented to the system by the Job Request are used to bind the remaining unbound parameters in the Job Description. The program information in the Job Description is formatted on the Task List and is forwarded, along with program control, to the Job Manager.

(2) Job Manager – The Job Manager is responsible for putting the job together and seeing that it runs according to specifications.

The Job Manager calls those programs which are given in the Task List, and gives them the required parameters from the Task List or from previous programs. The Job Manager continues to monitor the job until all tasks are completed. During this time, new tasks may be created by the Task Processor in the form of a Job Extension. When this occurs, new tasks are passed back to the Job Request Processor for inclusion on the Task List, and the cycle repeats itself. For example, a query job may call for a query translation and the query translation program may use a job extension when it has established the programs and parameters needed to execute the query.

(3) Task Processor — A task processor may be any of the job programs in the system, and it is responsible for performing the useful processing in the system. The job programs are executed under their own control. In addition, all accesses to peripheral storage through the LCP for the processing of the job itself are controlled by the task processors or job programs. Reference is made back to the Job Manager when the task is completed, when an unsolvable error condition occurs, or when a new task is encountered which requires an additional program.

3.4.5 System Programs

The programs in the system consist of two types:

(1) System-oriented programs are relatively stable, and provide the basic framework for processing data and for

implementing other programs. They constitute the underlying programming foundation upon which all other systems tasks are based. Examples of system-oriented programs are those which scan input data, which access and manipulate system directories, which locate data in the data base, and which do routine jobs such as sorting and merging.

(2) Task-oriented programs are components of user jobs. They use the system-oriented programs to help them accomplish the tasks required of the Reliability Central users. The task-oriented programs are less stable in that they may be changed to suit the needs of an evolving task or the user, and may be added to or deleted from with much less effect on the system than the system-oriented programs. Examples of task-oriented programs are those which summarize and analyze reliability data, those which generate specialized reports, etc.

The two types of programs are combined by a Job Description and work together to accomplish useful work. The data manager or reliability specialist responsible for accomplishing a task creates a Job Description through a Job Entry Request. Although many system-oriented programs concern themselves with system maintenance only, many others can be used as subroutines in user-oriented programs.

The use of the two program types may be illustrated by the following example. A reliability specialist may wish to extract some prescribed fields of data from a permanent file in order to build a temporary file more suited to his purposes. He may plan to perform a series of analyses on the new file and have the results reported in a

certain order and format. The specialist would issue a set of Job Entry Requests to produce the necessary Job Descriptions to tell the system what to do. The Job Descriptions would consist of a list naming both system-oriented and task-oriented programs. The system-oriented programs would scan the inputs which have been written in a user language, locate and extract data from the old file, construct the new file structure, and extract from the new file those data items desired by the analysis routines. If a generalized report generator were used, it too would be system-oriented in that it would be available to all system users. The task-oriented programs would permit the system to select the data according to the Boolean logical condition, summarize the data, analyze it, and prepare the desired output if a specialized report routine were used.

System-oriented programs which deserve special mention are the File Search and the Table Access Packages. These generalized programs are responsible for all basic data manipulations and searches. They use the directories for extracting, storing and altering data in the data base, and for restructuring data into a new form. One of the prime features of these programs is that they can make full use of the index data found in the Item Position Index, to perform conditional searches of the data base. Further, they can manipulate the index data and reduce the number of accesses to peripheral storage to the absolute minimum. This feature provides the foundation for accepting a query in any defined query language, and for processing it in an efficient manner.

SECTION IV. SYSTEM DESCRIPTIONS AND COMPARISONS

4.1 GENERAL

Eight data handling systems are described in this section with each system compared with the Reliability Central Data Management Subsystem. The systems have been chosen because they represent or illustrate significant contributions to the state of the art of data management. Several other systems were reviewed but were omitted from the survey for several reasons. The AIS file management system was excluded because the non-proprietary part of its documentation is insufficient to describe it in detail. The Army CCIS-70 project was not included because its programs do not have the same degree of generality found in the other systems. It is felt that the descriptions of the eight systems and RCDMS cover the major contributions to data management, and that inclusion of any further systems would be redundant.

No attempt has been made to classify the eight systems. Too many overlapping characteristics exist to make a classification meaningful. Rather, they represent a spectrum of achievement from the limited query capability of RECOL to the advanced data management capabilities of ADAM. The other systems can be placed somewhere in the middle of the spectrum. Their position in the spectrum depends very much on what system characteristics are important to the user, rather than upon some inherent ranking.

The authors of this report assume full responsibility for its contents, but they acknowledge that the system descriptions may contain certain inaccuracies, brought about by the following conditions:

(1) Most of the systems are in the design or development stage and are constantly being changed and improved.

(2) Due to restrictions of budget and time, some of the material has been taken from summaries and reviews rather than from definitive source material.

(3) The report completion schedule did not permit time for the system descriptions to be submitted for review to the designers of the systems treated in the report.

4.2 RETRIEVAL COMMAND ORIENTED LANGUAGE (RECOL)

4.2.1 Objectives

RECOL is the acronym for REtrieval Command Oriented Language. It was designed and implemented for an RCA 501 Tape System by W.D. Climenson in 1962. RECOL more properly belongs under the general heading of a General-Purpose Query System rather than a Data Management System, since it does not update files nor alter their structure. It is a generalized system in which queries can be posed to the system in a natural English-like language termed RECOL. RECOL is somewhat limited with respect to what can be achieved within the present state of the art, but its importance lies in the historical fact that it is one of the first general query schemes that utilizes a natural English-like language. Its basic objective was to provide a sophisticated means of interrogating a linear (i.e., tape) file which would permit the user to state a complex query conveniently, using any arbitrary combinations of the connectives AND, OR, or NOT with any degree of nesting. A further objective was to utilize a generalized program which would be capable of operating on any file which met the constraints of the system.

4.2.2 File Structure

The file structure utilized in RECOL is rather simple. The program can operate upon only one file at a time. A file is restricted to a single tape. Records within a file are unlinked. That is, one record cannot point to another record. The maximum record size has been restricted to 160 characters (or two punched cards) in length. Each record is subdivided into fields. The length of each field within a record is fixed. Each field may have a different field size. At the time of loading the file, a description of the record is provided which specifies the name of the field, its length, and the type of information contained in the field (e.g., integer, alphanumeric, floating point number).

The programs have been written around the data description so that many files having different formats may be utilized by the same program. The program does not permit data to be stored on disc. Data cannot exceed one tape in length.

4.2.3 Language

There is only one language permitted in the system. This is the retrieval language called RECOL. No maintenance commands can be specified. The user must maintain his own file prior to placing the data into the computer.

RECOL permits an English-like language expression to be presented to the computer. An interrogation consists of a group of orders to be executed in sequence. Five basic orders are permitted. These are SELECT, NAME, ASSOC, EDIT, and SUM.

(1) SELECT — The SELECT order retrieves data records specified by a logical condition. The logical conditions consist of the following:

Boolean Operators: AND, OR, NOT

Relational Operators: =

The logical condition is unrestricted by the number of Boolean or relational operators or by the nesting permitted.

(2) NAME — The NAME order permits a field to be named and a value to be placed in the field by stating a series of logical conditions. Thus, a new field can be created in a record. The logical condition is identical to that in SELECT.

(3) ASSOC — The ASSOC order permits records to be associated if they are related under certain conditions. The ASSOC order is specified by a set of maximum or minimum differences permitted between values of file record items. For each

record, the order will examine the remainder of the file to determine if any records satisfy these comparison limits.

(4) EDIT — The EDIT order causes a file to be sorted and printed (usually a file of selected, named, or associated records). The ordering of the printed data is specified by listing the record items from the most significant criterion to the least. No capability exists for specifying a logical condition and selection of a field based upon the logical condition.

(5) SUM — The SUM order counts records and finds average values. The EDIT order contains SUM.

The basic orders may be combined in various ways to form an interrogation statement. The orders NAME and EDIT may be cascaded. It was not possible to determine the precise syntax for utilizing orders, or all the ways in which the orders may be presented. Each command to the machine is preceded by a semicolon. Attributes are separated from values by a colon. A typical order might be

; SELECT · PAY-TYPE: WEEKLY, SEMI-MON, MON

The order name is separated from the attribute PAY-TYPE by an item separator. The attribute PAY-TYPE can take on any of the values listed and be selected. The values are separated by commas.

4.2.4 Implementation Aspects

A batch capability has been designed into the program to take advantage of a file record while it is in core. Up to five SELECT and/or NAME orders can be executed in a single pass. The logical conditions specified by the SELECT and NAME orders cause an executive routine to generate an object program to match the file records against the conditions. The logical condition is first scanned from left to right and translated into a Polish prefix format. The prefix form is scanned from right to left to generate a closed subroutine with true and false exits. The generated subroutines are added to a skeleton supervisory program which controls the input-output and record typing functions.

4.2.5 Comparison with Reliability Central Data Management Subsystem

As an early query system, RECOL met its own objectives with considerable success. As a total system, however, RECOL is not germane to the RCDMS objectives because it deals only with data retrieval. It does not provide a maintenance function or any of the controlling functions which comprise data management. Other aspects which make RECOL inappropriate for the RCDMS are the following:

(1) Because RECOL is tape-oriented, all query processing is performed serially, without the advantage of random access storage. RCDMS anticipates a short response time and must utilize random access storage at least for those items which are accessed most frequently.

(2) RECOL can operate on only one file at a time. RCDMS will often have queries and data analysis jobs that operate on several files. The RCDMS provides for access to any combination of files at any time.

(3) The files used by RECOL are in a frozen format. Once they have been defined to the system, they cannot be altered except by means of a separate file maintenance routine. RCDMS permits files to be redefined by means of a regular system job request.

(4) Each record in a RECOL file must have the same fixed format with no variable or optional fields. Files cannot be contained within files. The data processed by RCDMS is too varied for such major restrictions. The RCDMS directories are sufficiently flexible to permit almost any combination of variable fields, optional fields, and embedded files within the framework of only minor system conventions.

(5) RECOL contains no data indexing. Since RECOL searches serially rather than by using random access, it must scan each data record in the file to be certain that all pertinent data is retrieved. RCDMS requires data indexing in its directory because it uses random access storage devices. The index scheme employed permits the system to fetch from random access storage only those data items that satisfy the criteria of the query being processed.

Although RECOL is not germane to the Reliability Central as a total system, RECOL still contains several query aspects common to the Reliability Central objectives. Consequently, RECOL experience has contributed to the design of the Reliability Central Data Management Subsystem. Some of the RECOL conceptual achievements on which RCDMS designers have drawn are:

(1) The format of each file can be described to the RECOL processing program so that a single, generalized program

can be used to process all files. This concept is also a basic theme of RCDMS.

(2) A number of data manipulations are provided by RECOL, other than the selection of data by Boolean operators:

(a) Records may be matched and collated as determined by specified fields.

(b) Records may be sorted by a predefined sort key, and printed in a prescribed format.

(c) Data may be summed.

These functions, and several others, will be provided for in RCDMS by means of generalized system programs which can be sequenced and controlled by the user to meet his requirements.

(3) RECOL queries can be batched. With RCDMS, there is less need for batching because console and random access tend to dispose of the queries quickly enough. In the ultimate system planned for after the Test Operation, query requests will be organized for parallel processing by priorities.

(4) Queries can be posed to the RECOL system in a natural English-like language. Arbitrary combinations of the logical connectives AND, OR or NOT can be specified to any degree of nesting by the RECOL user. The program coding for responding to each query is compiled by the RECOL system at the time of input.

These three aspects of RECOL concern the query language and will not relate directly to the RCDMS until the query language for RCDMS is procured. However, these aspects are of great importance because the RCDMS must provide the query language with a compatible interface, and with a means of searching for data using complex logical conditions.

4.3 ACSI-MATIC COLLATION SYSTEM

4.3.1 Objectives

The ASCI-MATIC Collation System is an information storage and retrieval system designed and implemented by RCA for the Department of the Army, Office of the Assistant Chief of Staff, Intelligence. A prototype of the full system was implemented on a Sylvania 9400 Computer with a Data Products Disc File. Design of the system commenced in 1960-1961 and the prototype was implemented in 1963.

The objective of the information storage and retrieval system was to provide a common data base which could be utilized by intelligence analysts with different area responsibilities. The prototype system was to be designed to handle all the complexities of the full system with certain exceptions. The full system was designed to reduce the amount of effort required of analysts to evaluate, associate,and file new data. The latter function was to be achieved by automatically collating new information with previously stored information on the basis of information content. The ability to automatically collate new information was not part of the prototype system

At the time of the design of the system (1960-61) little previous work existed in this area. The system is of considerable interest since it is organized to utilize disc, and effectively accomplishes this objective (within the scope of the system).

The specific system of objectives for the prototype was as follows:

(1) Executive Control — The executive control system was intended to provide:

(a) Transfer of control between programs to take advantage of idle time between disc seeks.

(b) A centralized input-output package.

(c) An operating system which scanned inputs to determine the type of activity to run and to automatically retrieve and initiate the program.

(d) A dynamically allocated core memory (a trivial scheme was to be developed and should not be confused with the full system objective).

(e) A dynamically allocated disc storage.

(2) File Structure — The file structure was to be designed to:

(a) Facilitate data retrieval by indexing the data in great depth.

(b) Permit the retrieval of data which are related hierarchically.

(c) Permit the manipulation of lattice-structured information.

(d) Permit the addition of new terms, the deletion of old terms, and the addition of relationships of old terms.

(e) Effectively utilize disc to store the data and indexes to the data.

(3) Language to be Utilized — An English-like language was to be designed to:

(a) Query the system.

(i) Boolean 'AND' statements were to be permitted.

(ii) A specialized summary of data was to be permitted depending upon security.

(iii) Retrieval of hierarchically related data was to be permitted in the retrieval statement.

(iv) Print statements were to be permitted.

(v) Data was to be extracted from records satisfying queries.

(b) Establish new information

(i) ADD new data to a record.

(ii) DELETE data from records.

(iii) REPLACE data in records.

4.3.2 File Structure

To permit the rapid retrieval of data, the ACSI-MATIC file structure was patterned after a list type structure in which the record links were extracted from the data records and placed in a directory. Thus, the ACSI-MATIC file structure consisted of a directory (or thesaurus) and the data base. The unit of information in the data base was termed the Information Record of which there were to be different types such as Military Organization and Personality. Data entering the system might be pertinent to several such records. Each record type had a set of allowable information that could be stored in it. However, no data description language existed in the system so that frequent changes in system requirements meant program changes. Information Records of a particular type (i.e., Military Organization) could be linked to one another by containing an address in a particular field locating the linked record. Information Records of one type could also be linked to another type (e.g., Military Organization could be linked to Personality). Formation of a new Information Record could be achieved explicitly through an order (analyst order) or implicitly by the entry of a new message bearing information to the system.

The system directory consisted of two major entities: a Glossary and a Hierarchy Index List.

The Glossary served as a list of terms that were acceptable to the system. For each attribute class (e.g., Personality Name or Job Title), there appeared a listing of all values acceptable to the class, together with their encodings. The encoding expressed a hierarchical arrangement of terms within the class. Thus, a tree structure (and more generally, a lattice structure) of terms was permitted within the Glossary. The position of a term in the tree was specified by a binary code called Flexicode, which had the property that both vertical and horizontal coordinates were well specified. Further, given any two codes, it is possible to insert a new code between them at the same level. Codes could not be inserted conveniently between two levels. A special symbol denoted that a lattice condition was reached in the tree. Glossaries were maintained in alpha-numeric sequence.

The aforementioned Flexicodes were maintained in "hierarchical" order in the Hierarchy Index List. Each term in the list contained the Flexicode for the term it represented, plus pointers to the Information Records in the file containing information about the term, and to the glossary record. The pointer to the Information Record listed the physical address of the record on disc. It was found by the designers that a modification which would have listed logical names of records in a table consisting of logical names and their physical names would have saved considerable time in file maintenance since updating represented a significant portion of the system processing.

The directory (Glossary and Hierarchy Index List) was stored on disc. Since the Hierarchy Index List and Glossary could be quite long (extending over many

disc addresses) a method of focusing-in upon the term desired was developed. The technique involved utilizing subdirectories to the directory proper. For each Glossary or Hierarchy, the first word in the Glossary or Hierarchy of each group on disc is extracted from the Glossary and set up in a sub-directory. If the sub-directory extended over several disc accesses, it was subdivided in a similar manner. The main directory, which consisted of names of Glossaries and the name of the Hierarchy, pointed to the appropriate sub-directories.

4.3.3 System Processing of Messages, Orders, and Queries

The system inputs were designed to be independent of the input media. The system executive routine scanned the input media (paper tape, card) to determine if inputs existed. If an input was found, it was read into memory and scanned by a general-purpose scan routine to determine the input type. The following input types could exist:

(1) An input message.

(2) An analyst's order to operate on data records.

(3) An order to perform thesaurus maintenance.

(4) A query to the data base (in a complex manner).

(5) A report (several different special reports were possible).

Responses from the system were made on a monitor typewriter and a high speed printer.

The input language for message inputs, analyst orders, and queries was similar, while thesaurus maintenance orders had a somewhat different form. The language for the former utilized a unique name followed by a delimiter (a colon) to terminate the name (e.g., INTER: would represent an interrogation to the system). Following the delimiter was a series of tags and sub-tags representing several types of information. The first type represented auxiliary data such as the security of the input, the analyst's name,

the area he represented, the source document, and the date of entry. A tag was identifiable by terminating delimiter (e.g., : :). The tag represented an attribute. Following the attribute was a value. A second class of tags represented information about the message, order, or query. A tag (or attribute name) would have had several sub-tags belonging to it. Thus, for example, the tag PER for personality could have sub-tags for name, job, age, etc. Thus, a record could be considered to have sub-records belonging to it. In addition to tags representing attributes, other tags relevant to queries, such as subsumption; and to orders, such as to add a term to a record were permitted. Tags could be presented in any sequence and the system was capable of interpreting the input.

Messages were to be the normal method of adding information to the system files on a continuous basis. To enter a message, knowledge was required only of the message syntax and tags. System processing was to automatically associate data with other stored data, if pertinent, to automatically cross index the data, and to update the file. This feature was to be part of the extended system and was not implemented in the prototype.

The data base for the prototype system was to be entered using analyst orders. The orders were fully automated. That is, a new piece of data could be specified by an order, and the system set up the record, stored it on disc, and cross referenced the appropriate records in the directory. Orders permitted addition of new data, modification of an existing Information Record, and deletion of a record. A printout of the results was submitted upon completion. As a debugging aid, a monitor was designed to permit the system user to obtain a trace of the processing. Since several centralized packages were utilized (such as I/O, a thesaurus access package, and an input processor) it was found that unless one kept track of the processing, it would be exceedingly difficult to determine where errors had arisen.

Queries were designed to permit only AND operations for the prototype system. Apparently little consideration was given to OR and NOT operations and relational operators other than subsumption and equality. It is apparent, however, that other relational operators such as $\neq$, $\geq$, $\leq$, and the OR operation could have been added quite easily. Thus, one can consider that the prototype could have been modified with ease to permit Boolean statements consisting of a series of AND's followed by a series of OR's followed by another series of AND's in an alternating fashion. Queries consisted of an AND search criterion followed by an extract command and a print statement. Although a sort/merge routine was developed for the system, the scope of the prototype did not consider utilizing the sort/merge with queries and was to be part of the extended system. Basic retrieval operations involved the manipulation of indices rather than the viewing of all records during the data selection process. For example, consider the query, "List all physicists who are at a particular university." The job glossary would be retrieved to get the term physicists. The Hierarchy Index List would be entered to retrieve the list of addresses of all records containing physicists. The university glossary would be retrieved on the name of the specific university. The Hierarchy Index List for the university would list the names of all records containing the university name. Matching these two lists of addresses yielded the address of records containing both terms.

An input processor was developed to:

(1) Handle input format errors.

(2) Check for appropriate tags and sub-tags.

(3) Look up glossary terms to determine the system encoding.

(4) Resolve ambiguities (e.g., the location name NEWARK is ambiguous since many locations in many states can have the name NEWARK; however, if more information were provided on input, such as NEWARK, NEW JERSEY, the

processor could resolve the discrepancy to determine a unique encoding).

(5) Structure the data for further system processing.

4.3.4 Orders to Perform Directory Maintenance and Directory Use

The prototype system permitted several maintenance orders. The orders were designed primarily to add new terms to the Glossary and Hierarchy Index List. The addition of a new term meant that sections of the Directory had to be updated including the subdirectories an main directory. The orders utilized a parenthesized format. To add a term to a Glossary, or to modify an existing term, the name of the Glossary had to be given, the name of the term, and its hierarchic encoding. A hierarchy term to be entered into the system had to contain the hierarchic encoding of the term, the glossary term corresponding to the term, and information as to whether the term regularly was subsumed or subsumed other terms. The latter was needed in the event that the term was part of a lattice structure. A subroutine termed MAINT was utilized by the order to perform the physical maintenance of records and to maintain the directory.

To permit expansion of the prototype data base, space was left in the disc addresses assigned to glossary and hierarchy terms to minimize maintenance time.

A central package termed the Thesaurus Access Package (TAP) provided a means for accessing the data. TAP permitted files to be locked out while maintenance was performed, took advantage of the fact that a routine required sequencing through a Glossary or Hierarchy so that directory records already in core did not have to be reaccessed, and looped through the main directory and subdirectories to get at the appropriate terms. Programmers requiring use of the directory did not have to concern themselves with the structure of the directory and merely requested the pertinent information and were provided with the appropriate records.

4.3.5 Programming System

Several general-purpose utility routines were developed. The executive control routine was designed to permit transfer of control between routines, allocate core storage dynamically (a primitive storage allocation routine was developed), centralize input-output calls, and allocate disc storage.

The programming system for ACSI-MATIC was composed for an input-command package, an executive control package, an input-output package, and a collection of utility packages.

The input-command package accepted and interpreted commands for the execution of major system functions and for establishing the appropriate programs to effect the execution of those functions.

The executive control package for ACSI-MATIC furnished a collection of functions to enable the linking, loading, binding, and execution of system tasks. In addition, functions to enable parallel processing (up to sixty-four tasks) and dynamic allocation (i.e., requests for more space generated at execution time) were available. Basic blocks for loading, linking, and parallel processing control were kept in terms of chains of descriptions; i.e., lists of descriptive information were organized via chain links to reflect program relations, resource allocations, and priority for execution.

The ACSI-MATIC programming system furnished the user with a variety of utility packages. For example, requests for retrieval of items from the disc file by name were accepted and handled by the Thesaurus Access Package. In addition to retrieving data from the files by name via indices, this package furnished all of the machinery for adding and deleting entities of the files. Techniques for interlocking accessing during maintenance and for preserving accessed information in core for subsequent use were developed.

A second utility package was a comprehensive sort/merge package which was available to the user. This program allowed sorting on the basis of complex user-furnished ordering criteria.

A programming language and an assembly program were also developed. The assembly produced relocatable code and sufficient descriptive information to satisfy the requirements of the dynamic allocation and linking algorithms.

4.3.6 Comparison with Reliability Central Data Management Subsystem

The Reliability Central Data Management Subsystem has utilized several concepts and features in its design that appear in the ACSI-MATIC data retrieval and query system. In almost all cases, the concepts and features utilized have been generalized in order to increase the power and scope of their application. The following aspects of ACSI-MATIC have been used and expanded:

(1) The RCDMS file structure is a generalization of the ACSI-MATIC system. The Term Encoding Table of the RCDMS is equivalent to the totality of glossaries in the ACSI-MATIC system. The RCDMS directory will contain subdirectories, as in the ACSI-MATIC system, to permit one to focus in on the data rapidly.

(2) The concept of a centralized directory access package of the ACSI-MATIC system was a very useful feature which has been adopted in a more complex form for the RCDMS.

(3) The hierarchic encoding scheme, Flexicode, was a successful concept. The Item Class Code is its counterpart in the RCDMS.

(4) The advanced search strategies being developed for the RCDMS are outgrowths of the manipulation of indices performed by ACSI-MATIC.

(5) The concept of a generalized executive control routine has been carried forward to RCDMS.

Some of the aspects of ACSI-MATIC from which RCDMS departs in order to meet its own design objectives are given below:

(1) Data in ACSI-MATIC could not be restructured without extensive reprogramming of the system because no data description language was available. RCDMS has such a language and will permit logical restructuring of the data base.

(2) ACSI-MATIC did not utilize the concept of forming jobs from a series of generalized, task-oriented programs although a good measure of program generality was used.

(3) In ACSI-MATIC the Hierarchy Index List pointed to the physical address of the data rather than to a logical name for a record. This meant that moving a data record from one disc location to another required an excessive amount of time to be spent in performing maintenance on the directory.

In RCDMS, a table has been placed in the directory to translate logical names of data items into physical names. This feature separates the main directory from the physical address of the data, and significantly reduces directory maintenance.

4.4 NAVAL AVIATION SUPPLY OFFICE SYSTEM (ASO)

4.4.1 Objectives

The ASO (Naval Aviation Supply Office, Philadelphia) System is an inventory information retrieval pilot system designed by the Moore School of Electrical Engineering (University of Pennsylvania) under sponsorship of the Systems Research Division of the Bureau of Supplies and Accounts and the Information Systems Branch of the Office of Naval Research.

The ASO System design concept (executive routine, file structure, and query language) has been applied experimentally as a pilot model solution to the Sea Surveillance problem at FOCCLANT. The work on Sea Surveillance was done by Computer Command and Control Company for the Advanced Planning Division of the Office of Naval Research. The system description and conclusions presented here can, therefore, be considered applicable to the Sea Surveillance application.

The ASO is a supply control point responsible for the supply and inventory of items concerned with naval aviation. It presently maintains eleven major categories of files but the system discussed here is concerned only with a small part of one of them, the Master Data File (MDF). The MDF concerns some 460,000 inventory items to which there are approximately 190,000 changes made each month. Of the 460,000 items in the MDF only 4,100 items, less than 1%, are involved in the pilot system discussed here. Although the system is not a full-capability data management system, it is being briefly discussed in this survey because of the unique "multi-list" file structure it employs.

The objective of the ASO system is to furnish a user on-line query system for a random-access data file. Although the system is designed for technical, inventory, and contract management files, it is applicable to other management problems. The data in the files was to be accessible through many different categories of retrieval keys or combinations of retrieval keys in a logical AND/OR specification. The ability to enter the files by means of remote inquiry stations was to be provided.

A file maintenance capability was required that could accommodate approximately 10,000 changes in the technical data per quarter and 3,500 transaction changes daily in the file of 4,100 items. The information retrieval capability was to accommodate some 200 data requests per day. Periodic output consisting of 15 reports sequenced by a variety of keys was to be accommodated.

4.4.2 Implementation and Status

The present system consists of an IBM 1410 computer with two 1301 disk files, console, printer, tape units, and six IBM 1014 remote inquiry stations. The system is now operational at the ASO with a small portion of their data base, representing active, quick-response, high cost inventory items, on disk. The system described here supports a larger, conventional tape-oriented inventory control system.

4.4.3 Components

The system consists of a file structure, a query language, an executive routine, and various "worker" programs.

4.4.4 File Structure

The unique aspect of the ASO System is its file structure. There are two components to the file structure: a set of hierarchical index files and a data file. The data file is composed of records (sets of information relating to one object, for example, an airplane part, identified by its federal stock number (FSN)). The format

of a record describing one object is similar, but not identical to, the format of all other records. Each record contains values or descriptors (the smallest units of information that are of interest to the user). A value is a record for a particular attribute such as the number of parts in stock. The data file is stored sequentially according to values for one of the attributes (FSN) so that whenever interrogations and changes accumulate, routine processing can be facilitated by presorting the collected data according to that attribute. The file may then be treated as a single list comparable to the storage of information on magnetic tape.

An important feature of the system is the facility to enter the files by the specification of many different attributes. The way in which this is accomplished is the essence of the ASO system. It is accomplished through the use of hierarchical index files, called trees, and the use of a link field (address) associated with each attribute (other than the main sequencing attribute and other unique attributes) in every record. Because the link is the address of the next record with the same descriptor (value for the attribute), the data file is called the "multi-association" area.

There is one tree for each attribute in the system. The final level of the tree for each attribute is the list of values that the attribute can take (descriptors) and, for each value, a pointer (link) to the record address of the first record which contains that descriptor value. A link field associated with that attribute in the record contains the address of the next record with that descriptor (the link field is omitted for those descriptors known to be unique, such as FSN), and so on so that all records with that descriptor are on a linked list. A higher level in the tree serves as an index to the lower level when the value list is too large for one track. That is, it is a list of the last data value for each attribute that can be found on a given track with a link to that track.

4.4.5 Query Language

The query language in the ASO is process-oriented rather than user-oriented. A query is made up of a fixed sequence of nine numeric and mnemonic fields. The fields are numbered and a "/" is used as a word separator within a field. The format is as follows:

/// 1/ (process) // 2/ (key) // 3/ (conditions) //
4/ (elements) // 5/ (priority) // 6/ (security) //
7/ (requester) // 8/ (input identification) // 9/ (output identification) ///

Fields two through nine are optional. The nine fields have the following significance:

Field (1) Process. A four-character mnemonic for the process name.

Field (2) Key. A number indicating whether a single value, list, or range is desired, and the mnemonic of the key (FSN, FIIN, etc.).

Field (3) Conditions. A three-character numeric identifying the attribute, a single digit for the relation (equal, less-than, etc.), followed by the field values. A list of conditions which are interpreted as a logical AND criterion may be specified.

Field (4) Elements. A list of three-digit numbers identifying the elements to be output.

Field (5) Priority. A two-digit priority level.

Field (6) Security. A one-digit level.

Fields (7) through (9) are not implemented at present.

4.4.6 Executive Routine

The executive routine stacks requests and executes them according to priority. Since the computer is not capable of parallel processing, the executive routine is not designed to cope with such a facility. Requests may be made from the console or a remote inquiry station. Answers to a query are made to the unit which is making the query. There is no real-time interrupt but programs are written so that they transfer to the executive routine quite frequently (e.g., twice per second). The executive routine then checks to see if there is a service request at an input device. A low-priority query program will be interrupted and saved to permit a high priority query to be answered.

4.4.7 Comparison with the Reliability Central Data Management Subsystem

The ASO system is a technically significant subset of a file management system. However, in its present form the design is not directly applicable to the objectives of RCDMS for the following reasons:

(1) The input capabilities of the ASO system do not provide a job specification language or a data description language. The wide variety of jobs and data structures in the Reliability Central operation require that RCDMS have both types of languages.

(2) The ASO query language requires strict sequencing of tagged fields. When preparing a query, recourse

must be made to a handbook of codings for attributes, descriptors, relations and the like, as well as for the format of the query itself. The objective of the Reliability Central is to give the user a more flexible, less demanding query language.

(3) The search strategy of the ASO system employs the list structure method. Similar attributes in the data base are linked by addresses in the data record. This method results in many accesses to disc while following the chain of linked data attributes. RCDMS has, in effect, taken all the link addresses out of the data records and put them in a list form in the directory. The RCDMS search strategy is, first, to manipulate these lists at high speed in core storage to determine the addresses of all data records which satisfy all the query criteria. Only the pertinent data records are then retrieved from disc, eliminating all needless accesses.

(4) The ASO system data structure is limited to a single level file with no nested or embedded structures. This restriction reduces the ease of user communication in problem areas where the data structure is naturally complex. The Reliability Central data base will require the flexibility afforded by the embedded structures in the RCDMS design.

The "tree area" of the ASO system directory has a structure which is quite similar to the RCDMS directory. The system keeps track of data names and groups by means of a hierarchy of indexes which can be accessed quite efficiently. Thus, it is in the data structure that searches become inefficient (as discussed in Item (3) above) because the links are contained in the data records instead of in the directory. Accessing specific data values involves chaining through links contained in the data itself. Only the first link of each chain is kept in the ASO system directory.

4.5 INFORMATION PROCESSING SYSTEM (IPS)

4.5.1 Objectives

IPS (Information Processing System) is a data management system designed by NAVCOSSACT (Naval Command and Control System Activity) and implemented by IBM for a multi-computer environment at FOCCPAC (Fleet Operations Control Center, Pacific).

The objective of IPS is to provide a comprehensive file query and maintenance system for a command and control center user/programmer staff. Specifically, the functions that IPS had to serve are as follows:

(1) Data used by command staff personnel had to be structured into a data base for use of such functions as operations, intelligence, logistics, and administration.

(2) Data base maintenance and recall of information had to be accomplished easily for data structure which may consist of up to 200 files.

(3) The system had to be used and maintained by staff personnel, programmers, and operators.

(4) IPS had to be programmed completely in JOVIAL.

(5) Its general capabilities had to be compatible with Navy command system data processing techniques.

4.5.2 Implementation and Status

IPS has been implemented for a CDC 1604-A in the FOCCPAC multi-computer system officially designated as AN/FYK-1(V). The Phase II system, presently operational,

is a tape-oriented system. Both the IPS system and the data files are operated from tape. A disc system is currently being proposed.

The implementation language is JOVIAL. Programs in the IPS language run in the interpretive mode within the IPS executive.

IPS is a generalization and refinement of the TUFF (Tape Update for Formatted Files) system, developed by IBM for the Operations Research Division (ORD) of the David Taylor Model Basin.

IPS provides the following capabilities not possessed by TUFF:

(1) The ability to represent and order repeated items within repeated items (nesting several levels of files within records).

(2) The ability to define two or more sets of repeated items (files) at the same level within a record.

(3) A query language for querying any data base file.

(4) A problem-oriented language for describing file maintenance operations.

4.5.3 Components

IPS is divided into three major subsystems, the File Maintenance System (FMS), the Information Retrieval System (IRS), and the Library Maintenance System (LMS). FMS is used to generate and maintain the formatted files used by the system; IRS is used for the selective retrieval of data from these formatted files; and LMS is used to generate and maintain a file of commonly used tables which are always available to the system at "run-time." The tables include the so-called "macro-list" for each data file in the system. The macro-list is a program in a "problem-oriented" file

maintenance language which updates a particular file in the system. (Each file has a section of the library tape which has all tables concerned with that file.) The macro-list is executed interpretively by the IPS executive during the update run.

4.5.4 File Structure

The basic IPS philosophy is that all operational data requirements of the operations control center can be organized into a set of fundamental structures called files, where each file exists as a tape reel or series of tape reels (multi-reel file), and that the structure of each file may be defined in a File Format Table, maintained as a record on a library tape.

Each file contains four basic types of items:

(1) Fixed length non-repeated items.

(2) Variable length non-repeated items.

(3) Fixed length repeated items.

(4) Variable length repeated items.

Each operational area in the operations control center defines its data requirements explicitly in this form and develops files for its use.

Each item in a file structure is identified in the File Format Table as to its type and at what level in the structure it appears. A file structure may contain imbedded files, called repeated groups, up to 64 levels of nesting. A group of items in a record (whether repeated or not) which has a name is called a "set."

A record in a hypothetical file on Naval Fleet organization is illustrated in Figure 4-1. Each record in the file would represent one fleet and would contain the same logical (although not necessarily the same physical) structure. In Figure 4-1, Fleet A is

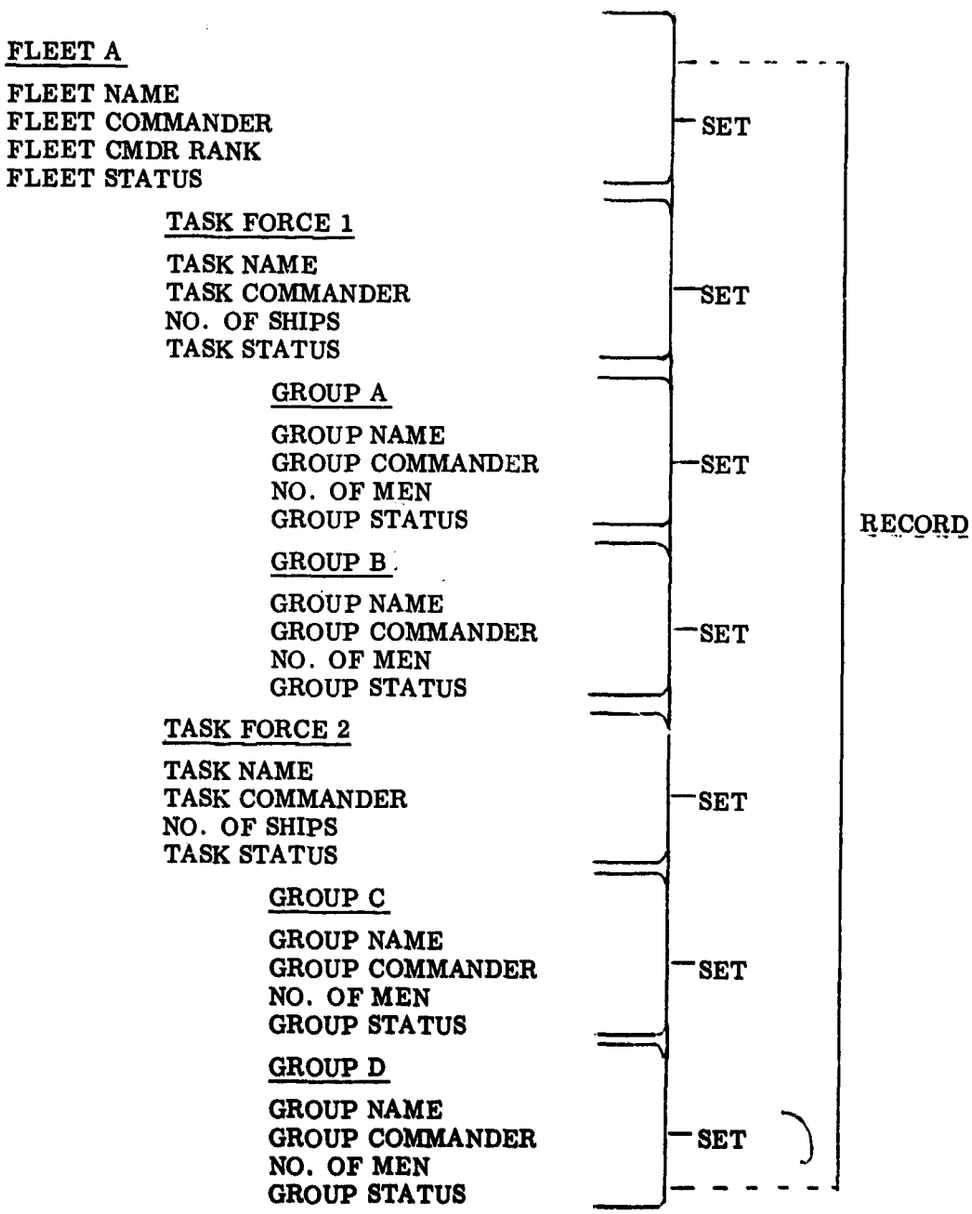


Figure 4-1. Outline of One Record of a File

a set and consists of the group of items entitled Fleet Name, Fleet Commander, Fleet Commander Rank, and Fleet Status.

The Information Processing System requires that all sets within a record be assigned a repeat level number. This level is determined by (1) the number of times certain sets of items may be repeated within a record and (2) how far up the logical ladder the set is located from the principal set of the record.

In Figure 4-1, the items listed under Fleet may appear only once per record, and will constitute the principal set of the record. This set will therefore be assigned a repeat level number of zero. The sets entitled Task Force 1 and Task Force 2 are at the next repeat level (number one) because as sub-headings of the principal set Fleet, they are one step up the ladder. Similarly, Groups are another step removed from the principal set and are given the repeat level number of two.

There may be as many as sixty-four repeat levels per record, numbered zero through sixty-three. All of these repeat levels, with the exception of zero, may appear more than once in a record but the zero repeat level must appear only once per record. If an item occurs more than once within the same record, it is known as a repeated item. In Figure 4-1, the items listed under the set Task Force are repeated items because they appear more than once in the record. Similarly, if a set appears more than once within a record, it is called a repeated set. This means that Task Force is a repeated set and Group is a repeated set within a repeated set.

4.5.5 File Maintenance System (FMS)

The File Maintenance System is an executive system which, at the time of an update or generation run, reads the file format table and macro-list for the particular file involved, and executes the macro-list in the interpretive mode.

The inputs and outputs of the File Maintenance System vary according to the user's needs. The major inputs are the old Master Data File, the Transaction Data File, the Library File, and the Card Input File. The major outputs are a new Master Data File, two optional Summary Files, and an Error File.

The Master Data File is the primary output of the FMS and the primary input to the Information Retrieval System. When the old Master Data File is being updated and corrected, it will also be input to FMS.

The Transaction Data File contains the information that is used as input for generating a new Master Data File or for updating and correcting an existing Master Data File. The Card Input File contains the run control card and other data for the particular run.

The Macro-List is a program which describes to the system how the data is to be manipulated for the update run. The complete Macro-Instruction Set for FMS is shown in Table 4-1. Macro-instructions used by FMS are divided into four basic categories:

(1) Environment establishing instructions, for initialization.

(2) Data handling instructions which manipulate the data.

(3) Control instructions for branching.

(4) Output instructions.

4.5.6 Information Retrieval System (IRS)

The goal of the IRS is to provide for the convenient retrieval of data from the Master Data Files. Requests are submitted to the system by the user in the IRS language.

MACRO-INSTRUCTION SET

OPERATION	FIELD A	CONNECTOR	FIELD B
MOVE	I, V	TO	I
COMPARE(1)	I, V	TO	I, V
LOOKUP	I	IN' TABLE	T
STORE' IN	I	-	-
MOVE	I, V	INDIRECT' TO	I
ADD	I, V	TO	I, C-REG
SUBTRACT	I, V	FROM	I, C-REG
MULTIPLY	I, C-REG	BY	I, V
DIVIDE	I, C-REG	BY	I, V
GOTO	O	-	-
IF 'X' GOTO(2)	O	-	-
STOP	-	-	-
STOP 'AND' WRITE	-	-	-
LOG' ERROR(3)	-	-	-
LOG' COMMENT	V (NOTE 4)	AND (OPTIONAL)	I (OPTIONAL)
WRITE' SUMMARY	-	-	-
WRITE' SUMMARY' 2	-	-	-
CLEAR' SUMMARY'	-	-	-
CLEAR' SUMMARY' 2	-	-	-
PRINT' SUMMARY'	-	-	-
PRINT' SUMMARY' 2	-	-	-
INITIALIZE	S	EXIT	O
POSITION	S	EXIT	O
STEP	S	RETURN' TO	O
BUILD	S	-	-
DELETE	S	EXIT	O
GENERATE(5)	-	-	-
CHECK VALID(5)	-	-	-

TABLE 4-1. MACRO-INSTRUCTION SET

LEGEND

I = Item name (14 characters maximum)
No * (prefix to item name) = Master Item
* (prefix to item name) = Transaction Item
** (prefix to item name) = First Summary Item
*** (prefix to item name) = Second Summary Item

S = Set name (8 characters maximum)
No * (prefix to set name) = Master Set
* (prefix to set name) = Transaction Set
** (prefix to set name) = First Summary Set
*** (prefix to set name) = Second Summary Set

O = Operation statement tag

T = Table name

V = Value; values are enclosed in ( ); 120 characters maximum

NOTES

(1) Both fields cannot be values

(2) 'X' settings

Equal	Greater
Nequal	Valid
Less	Nvalid

(3) Prestored system comment written on error tape

(4) Comment written on error tape

(5) Requires a function call in variable field

TABLE 4-1. MACRO-INSTRUCTION SET (Contd)

The system analyzes the request, finds the information, and prints it in a predefined, user-specified format. Successful use of the system depends upon: (1) a thorough understanding of both the data structure and the content of the files used, and (2) an understanding of the proper use of the query language.

A query consists of a group of statements containing three parts:

(1) the identification,

(2) the logical search condition, and

(3) the output format.

The logical search condition handles the full Boolean range of AND, OR, and NOT conditions on the items of a master file. The possible operands are item names, functions, and values. A function consists of a function name and a group of items and/or values which serve as inputs to a subroutine. It is used to compute a value not contained explicitly in the data. Each term in the logical search condition consists of a left operand, a relational operator, and a right operand. Relational operators include the usual algebraic relationships, and also directional operators, North, South, etc.

Every query must contain a request for a specific form of output. The type chosen by the user depends on whether or not he desires a report which is a straight listing of retrieved data or a summary of retrieved data. Four standard outputs are available from the IRS: list, column, sum, and count type edits. A sort option is available in a list-type output.

4.5.7 Library Maintenance System (LMS)

The function of the Library Maintenance System (LMS) is to provide the means for the generation and maintenance of the IPS Library File. LMS performs this function by accepting tables in the form of card decks, processing each card deck to produce

a particular type of table, and placing each table in the IPS Library File in such a manner that the table may be retrieved when needed.

The IPS Library File is made up of many segments of data on magnetic tape. Each segment is uniquely associated with one particular Master Data File and consists of a series of queries, tables and macro-lists which IPS will use in processing that particular Master Data File.

Each segment is divided into blocks of data which contain one query, table, or macro-list that can be used in generating and maintaining the same Master Data File. The blocks of data within the segment are always placed on tape in the same sequential order with the first block containing the information necessary for locating all other blocks of the segment.

The blocks of data within each segment are in the following order:

(1) Segment Index Block — This block contains the information necessary for locating any table in the segment.

(2) Library Queries — These prestored card images of information retrieval queries are the only tables on the Library File used by IRS.

(3) Summary Format Tables

(4) Transaction Format Tables — Used by FMS

(5) Macro-Lists

(6) Code Conversion Tables

4.5.8 Comparison with Reliability Central Data Management Subsystem

IPS provides an extremely flexible system for performing serial maintenance and retrieval functions on a large number of tape files, but it does not provide for the efficient communication of data between files, or even within one file. Therein lies one of the basic differences between IPS and RCDMS. RCDMS treats the entire data base as a single entity, using an integrated directory to describe all data. RCDMS makes a minor differentiation between files at the logical name level rather than a major differentiation at the physical address level.

Other differences are noted below:

(1) IPS is unable to pose queries that involve more than one file, or to pose queries without knowing the particular file involved. This constraint arises from the fact that the term dictionary for each file is located in a separate segment of the Library File, with no single master list of all system data terms. This forces a deep-nested data structure in certain files. RCDMS, on the other hand, uses a common data base approach which keeps all system terms and format tables in a combined directory structure, thereby allowing queries to involve any element in the data base.

(2) Because it is a tape-oriented system, IPS consumes considerable operating time scanning tape. It needs no indexing scheme because it must look at each record during its serial search. In contrast, RCDMS uses an extensive data indexing capability to maximize its random access potential for retrieving data rapidly.

(3) RCDMS provides job definition and job management aspects which are not included in IPS.

(4) Since programs in the IPS language run in the interpretive mode, they require more running time than if they had been compiled before hand. RCDMS updating functions will generally be performed with compiled routines.

Several features of the RCDMS have been drawn from IPS experience. In the following areas, RCDMS is indebted to IPS for conceptual development, design techniques, and approaches to implementation:

(1) IPS provides for a flexible logical data structure and physical storage format. The ability to nest repeated groups and to have any number of repeated groups in a record satisfies most of the basic user needs for individual data objects. RCDMS provides for the same flexible structure, as well as for cross linking of files and records through a directory-contained link system.

(2) IPS uses a term dictionary and format tables to control its generalized system programs for flexible query and update operations. RCDMS uses much the same concept.

4.6 HQ, USAF COMMAND & CONTROL SYSTEM (473L)

4.6.1 Objectives

The Headquarters, USAF Command and Control System (473L) is a file maintenance and retrieval system whose basic function is to assist the USAF staff in planning and monitoring military operations. It was designed and implemented by IBM for USAF ESD.

Like other command and control systems of this nature, the HQ, USAF system has mainly been oriented toward the command staff user rather than toward the ADP specialist. The system must permit the command staff user requiring data to interact with the computer and its data files without the assistance of a systems specialist.

There are three classes of users, each placing a somewhat different requirement on the user language: (1) the Air Staff Officer, who requires a simple, straightforward language, (2) data control personnel who, through daily use, can be expected to master the language, and (3) the system programmer to whom the degree of difficulty is secondary to the power of the language. A language was therefore required, the complexity of which would be a function of the requested action. For example, the Air\Staff Officer should need to know only a subset of the total Query Language, since he desires a fairly straightforward retrieval.

Due to the highly dynamic, unstable nature of the data base, the following two requirements had to be satisfied: (1) programming tools had to be available to the system programmer which would permit him to access data without tying his program to the data files as they existed at the time the program was written, and (2) there had to be a means of retrieving and processing data from the time it was loaded into the system data base, without waiting for a new program to be designed, coded, and debugged.

4.6.2 Implementation and Status

The Operational Training Capability (OTC) phase of 473L was implemented for an IBM 1401 system. The entire OTC system was upgraded to an IBM 1410 in June, 1964. A second phase, called the Initial Operational Capability (IOC), involving a Librascope L-3055, has recently been installed. A third phase, complete Operational Capability (COC), is scheduled for mid-1965.

4.6.3 Components

The L3055 computer has input-output devices that operate in parallel with the central processor, and an interrupt feature that allows inputs via communication consoles and data links. The equipment includes a disc which features a search-by-content accessing scheme, and fixed read-write heads which have an average access time of 35 milliseconds. Its capacity is approximately 40 million characters.

The communication console has a typewriter input keyboard, a set of 30 control keys that may be identified with one of 64 unique overlays, typewriter, and other displays.

In the 473L programming system, the ability to perform a distinct data processing task is called a capability, and the set of computer programs which makes a capability possible is called a capability program.* All portions of a capability program are stored on disc and are called upon by actions taken at a communication console.

*This is called a user task and a user task program in RCDMS, and is called a specific model and model specific program in ADAM.

The execution of capability programs is under control of a set of programs called the System Monitor. The System Monitor consists of an Executive Control Program, a Console Processor Program, and a Loader Program. The System Monitor receives all inputs to the system, determines whether a particular capability program is required, calls capability programs and monitors their operation, and receives an indication that a capability program has completed its task. Monitoring the operation of a capability program means: (1) executing the interpretable steps of a program when it is communicating with the console operator, (2) recognizing when the program requires working storage allocation or other service, (3) calling service or working programs and giving them control, and (4) accepting processing requests for I/O.

4.6.4 File Structure

The files in the OTC system were set up in parallel and the language was designed to operate solely on parallel files. In a parallel file the values for an individual property of all objects (records) are stored together, as opposed to a serial file in which the values for all properties of an object (record) are stored together. Thus, in a parallel file, each property is associated with a set of entries, one for each object the file describes. The nth entry of each property value list would be an item of the nth record of the corresponding serial file. Parallel files are inherently limited to a single level structure. Although parallel files made short data retrievals extremely rapid, they created a very time-consuming file maintenance procedure. In the IOC system files are serial-parallel: serial in the sense that all properties, or attributes pertaining to one entry, are stored together in the file; and parallel in the sense that an entry may have a primary section and a secondary section. These sections are stored separately, with each carrying an address connective to be used by the Query Language in maintaining the relationship between sections.

Within an entry, values may be carried in four ways:

(1) As a single value where only one value is possible for an attribute in an entry. This requires a fixed space for each entry.

(2) As multi-values in which each entry carries a fixed number of values in an entry. This also requires a fixed space in the entry.

(3) As multi-values in which each entry carries a variable number of values in an entry. This requires only the space necessary for the actual number of values possessed by that particular entry.

(4) As remarks or free text on an entry basis.

4.6.5 Query Language

The Query Language itself consists of a vocabulary and a grammar. The vocabulary consists of two parts, a fixed part and a dynamic part. The fixed part consists of unique words which describe and control the retrieval process. The dynamic part of the vocabulary contains words that describe the data to be retrieved, that is, the file indicators, attribute names, etc., of the data base. This portion of the vocabulary is automatically changed by the File Maintenance program any time a file in the system is added or modified.

The grammar of the Query Language is implemented by a syntax and a punctuation set. The syntax refers to the arrangement of words into statements which are meaningful

to the system. The syntactical relationships are specified by the punctuation and the unique words. A simple, uniform, English-like syntax is utilized to make the language easy to learn and retain. The punctuation set is also simple and uniform. Only as many marks as are required to clearly separate the elements are used.

There are seven basic statement elements of the 473L Query Language. In order of presentation and common usage the seven elements are the Program Indicator, File Indicator, Qualifier Conjunction, Qualifier, Selector Conjunction, Output Director, and the Output Selector. In certain Query Language statements some of these elements are omitted and, in very complex query statements, the elements may be difficult to recognize. Careful analysis, however, will reduce any Query Language statement to these elements.

(1) Program Indicator — This is the first word of a Query Language statement. It directs the system control program to use the Query Language program. It also provides a logical English language beginning for the statement. The word RETRIEVE is a program indicator, for example.

(2) File Indicator — This identifies the file from which data is to be retrieved and always follows the program indicator, for example, a file name like FORCE STATUS.

(3) Qualifier Conjunction — The word WITH follows the file indicator. It serves as a conjunction between the file indicator and the qualifier. It makes the statement more readable and precludes the need for punctuation to identify the beginning of the qualifier.

(4) Qualifier — This is the element of the statement which describes the specific nature of the data to be retrieved. A qualifier consists of a set of one or more modifiers, each of which is normally composed of an attribute, a comparator, and a value. An attribute is a characteristic of the file; a value is one of the states an attribute may assume; a comparator defines the logical or mathematical relationship between the attribute and the value. RUNWAY LENGTH is an attribute of the sample FORCE STATUS file and 5,000 feet could be a value for RUNWAY LENGTH. The expression "RUNWAY LENGTH > 5,000" is therefore a valid modifier. Another modifier could be "COMMAND = TAC." Placing these two together as "COMMAND = TAC, RUNWAY LENGTH > 5,000" forms a modifier set that describes certain entries in the file more specifically. The modifiers in a set are separated by commas and are logically additive; that is, an entry must meet the requirements of all the modifiers in a set to qualify. A simple qualifier contains only one modifier set. A compound qualifier may be constructed by combining several alternative modifier sets. This could be: "COMMAND = SAC, ACFT POS > 10; COMMAND = TAC, RUNWAY LENGTH > 5,000." The semicolon (;) defines the end of one modifier set and the beginning of the next. It also specifies a logical OR relationship between the sets. Data may qualify by meeting either of the modifier set's criteria.

(5) Output Director Conjunction — The word THEN always follows the qualifier. It serves to make the statement more meaningful from an English language view and acts as the separator between the qualifier and the output director.

(6) Output Director — This is normally a single word that specifies the output device desired and the format in which the retrieved data is to be presented.

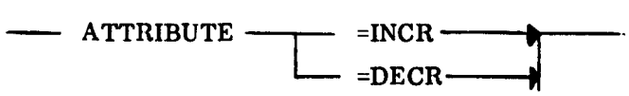
(7) Output Selector — The last part of a Query Language statement is the output selector. It contains the names of the attributes that are to be presented in the output and specifies the detail arrangement of the output within the general format specified by the output director.

An operator who desires to reuse a Query Statement may, at this point in the Query Statement direct that it be added to a SAVE table without processing, by adding the word SAVE at the end of the statement. He may also append remarks to the statement at this time.

A Query Statement is terminated by an End of Message symbol which is available on the 473L equipment.

(8) Output Options — There are two optional features available which the operator may use in conjunction with the OUTPUT DIRECTOR and the OUTPUT SELECTOR to order his data and to title his report. The sequence of attributes in the

output is controlled by the order in which they are listed in the statement, and the values are normally listed in the sequence in which they are retrieved from the file. To develop a more useful output, the operator may specify that the data is to be sorted. The expression for SORT is either INCR, increasing (i.e., A to Z alphabetically or in increasing numerical order) or DECR, decreasing, which is the converse. Sorting instructions can only appear in the output selector portion of the statement and are expressed in the following form.



Sorting may be performed on a number of different attributes in one statement. The attribute listed first will be the primary sort, the attribute listed second is the secondary sort, etc.

The other option in the output is the ability to give any QL retrieval a title by enclosing the desired title between asterisks. The title must be entered following the OUTPUT SELECTOR but preceding the terminal punctuation.

(9) Functions — In addition to the basic elements and options previously discussed, there are a number of special functions in the Query Language that may be expressed in the qualifier and/or selector portions of a statement. These functions provide the ability to qualify on or to generate and select

data based on data criteria not explicitly stored in the file. The functions are: Great Circle Distance (GCD) that computes the distance between two geographic points; SUM which accumulates the sum of several values of one existing attribute; and the MIN/MAX function which selects the minimum or maximum value among several attributes on an entry basis.

(10) Complex Queries — An additional feature of the Query Language, the complex query, provides the user with a powerful tool. A complex query provides the ability to develop a statement that is composed of several subordinate queries. Each subordinate query is separated by a colon (:) and can address the same or a different system file. The complex query retains data retrieved by prior subordinate queries as values of modifiers. Essentially it provides the ability to use data from other files to qualify and select data from one file.

Assume it is desired to retrieve all SAC units from the sample file that possess at least as many aircraft as the 413ATW at DENVA. This could be performed by using two separate statements by first retrieving the number of aircraft possessed for the 413ATW, then using the retrieved value in the modifier of another statement to qualify those units meeting that requirement. However, one complex query can perform the same task.

```
RETRIEVE FORCE STATUS WITH COMMAND =
SAC, UNIT = 413ATW THEN RETAIN ACFT POS:
RETRIEVE FORCE STATUS WITH COMMAND =
SAC, ACFT POS>  [R1, ACFT POS, OR]  THEN
LIST UNIT, ACFT POS
```

The above complex query would retrieve the value for ACFT POS for the 413ATW in the first subordinate query and automatically insert it as a value to the ACFT POS modifier of the second subordinate query. The second subordinate query would retrieve all units meeting the desired requirements. The colon (:) is used to separate the subordinate queries, and the information enclosed within brackets directs the QL in locating the data to be used. The R1 denotes the working file, the attribute ACFT POS denotes the attribute values to be used, and the OR denotes the relationship of those values when using them in the statement.

All previous discussions have related to the IOC Query Language. Additional features being planned for in the COC Query Language include a means of expressing mathematical computations that can be used in either the qualifier or selector portions of a statement. This will be performed by a planned COMPUTE function. The capability to extract information from several files to generate a new file that can then be queried will be performed by a planned COMBINE function.

4.6.6 Capability and Use

The man-machine interface in 473L is satisfied by several communication consoles from which operators may enter queries and view replies. A data link permits

remote stations to send messages, status reports, and inventories directly to the computer. The information received over the on-line data link is used to update the data files which are stored on disc. Many users have simultaneous access to the system files through the use of communication consoles. Access can either be in the form of a typed query statement or an option selected with the aid of a control panel overlay that defines a specific capability area. A combination of these two procedures can be used; for example, an overlay may be used to help an operator construct a query. Thus the control panel and the query language are complementary, and in the operation of the console they can be used alternately without interference.

4.6.7 Comparison with Reliability Central Data Management Subsystem

The HQ, USAF system has successfully solved the problem of providing a command staff with an optimal man-machine interface for on-line, non-planned queries. In achieving this, some generality has been sacrificed, as described below.

(1) The 473L system does appear to contain a mechanism for quickly defining and incorporating into the data base a highly structured data set. For example, there appears to be no capability in the serial-parallel file organization for representing an arbitrary number of embedded files in an entry. (This is the capability called "repeated groups" or "nested repeated groups" in NAVCOSSACT IPS). RCDMS uses its data description language, its Item Definition Job Request, and its Data Entry Job Request to permit the user to specify and incorporate data structures with any degree of file nesting.

(2) The literature does not speak of a capability to cross reference items within the 473L data base. This limits the system's ability to give complete answers to queries which refer to data in several different file sections of the data base. The RCDMS directories are designed to contain tables of link addresses from records in one file to records in other files. The cross reference can be established rapidly, because the links are in the directory, and the records of the file originating the link need not be accessed.

(3) The 473L system does not have a data indexing scheme. Consequently, query searches involve looking at the data rather than manipulating directories. The parallel organization of the files helps to reduce the amount of data handling in the searches, but the number of accesses to data is more than if the data were indexed. RCDMS provides for full indexing of the data to whatever depth the user desires. The search strategy of RCDMS is to make no access to peripheral data storage until the name of a specific data item has been isolated which satisfies the query criteria. The strategy is designed to eliminate all unnecessary data accesses.

(4) The ability to store and automatically trigger a complex sequence of data processing tasks as a production capability has not been designed into the 473L system. One of the features which makes RCDMS a general-purpose system

is its ability to receive and store a variety of Job Descriptions which can be run at any time in the future without reprogramming. Further, the user can introduce new parameters into the programs specified by the Job Description when the time comes to run the job.

(5) The 473L system does not seem able to enter new data descriptions in a user-oriented language, whereas RCDMS incorporates a flexible data description language which can be used in conjunction with an Item Definition Job Request to make changes in the logical structure of the data base.

The two systems have several objectives and design features in common, namely:

(1) Both systems are designed to ultimately serve a user population which has little or no knowledge of system operations, and yet which wishes to use the system directly, without a specialist acting as an intermediary.

(2) Both system designs strive to handle a dynamic data base by means of generalized programs wnich do not depend on any particular data structure.

(3) Both system designs permit a query to be applied through the whole range of the data base, thereby broadening the scope of the queries, and increasing the usefulness of the data base.

(4) The 473L system has a unique display capability which permits a broad variety of output formats for the user. When the RCDMS development reaches the stage where the display capability is being designed, the RCDMS designers will look carefully at the 473L design.

4.7 COMPILE ON-LINE AND GO SYSTEM (COLINGO)

4.7.1 Objectives

COLINGO (Compile On-LINe and GO) is a general-purpose storage and retrieval system developed for the USSTRICOM Interim Command and Control System by the MITRE Corporation, Bedford, Massachusetts.

COLINGO was designed to provide a computer-based system for data file generation, maintenance, and retrieval in accordance with uniform but flexible procedures.

It was to be a system that could grow and be easily modified based on the users' changing operational requirements, and the designers' technical experience.

These objectives and associated constraints dictated the following general design goals for COLINGO — COLINGO was to be:

(1) A system that would accept most types of data from most sources, with minimal reformatting.

(2) A system that would perform almost any logical or mathematical manipulation of the data.

(3) A system that would allow rapid updating without imposing difficult constraints on the update format.

(4) A system that would permit on-line programming to meet many day-to-day problems, and off-line programming for more complex problems.

(5) A system with simple program maintenance and extension capabilities to permit the using command to participate and guide the evolutionary design of the system.

(6) A system with a good man/machine interface and control language, which would be data-compatible with most associated organizations.

4.7.2 Implementation and Status

COLINGO consists of modular computer programs for an IBM 1401 computer. Control of the data, computer programs, and equipment is achieved through on-line interpretive execution of statements in the COLINGO Control Language entered by cards or from a single console typewriter. Programs are constructed as closed subroutines which may be written in Autocoder, SPS, or COBOL. Most system files are on tape, while programs and high priority data are on disc.

A COLINGO "C" Version, handling query aspects, was completed and checked out in early 1964. COLINGO "D" which implements additional control features, report generators, and sort routines was scheduled for completion in mid-1964. A COLINGO 10 is proposed for the IBM 1410.

4.7.3 Components

The system languages consist of a user language, the COLINGO Control Language (CCL), and a programming language approaching the capability of COBOL. All user communication with the computer is accomplished with a CCL message. The control language, although it contains a basic query language subset, is more than a

query language in the usual sense since CCL messages initiate all system actions including job execution and control over data, programs, equipment, and output. A typical message is shown below:

GET A-FILE IF STRENGTH/AUTH GR 500
EXECUTE 01 02 03 IF/NOT PRINT ALL.

(Underlined words are action verbs — all words following an action verb up to the next action verb are parameters of the next action verb. The parameters following the EXECUTE action verb are the labeled locations of other messages. Underlining of action verbs is for clarity only; they are not actually underlined in the message entry).

COLINGO consists of an Executive Program, a CCL interpreter, and a system of subroutines. It includes no inherent data base or operational capabilities. Its programming language and compiler are applicable to almost any data management task, but are entirely dependent upon the data base specification and programs written for the particular problem to be solved.

System macros provide a capability for the following action types (each represented by several macros), used at the CCL level.

Input/output
Data manipulation
Computation
File generation
Update and maintenance
Sequence control

A data description language is provided (the COBOL "PICTURE") within the programming language. Dictionaries are generated by the system to be used by the data management routines.

4.7.4 Data Structure

The format of a record can extend to two levels; that is, an item may have one subitem or one parent item (but not both). The formats of the logical records in a given file are not all necessarily the same, since an item may be a subfile. The program structure provides for up to one subfile by providing one "master" physical record plus up to fourteen "trailer" physical records for each logical record; the trailer records provide for values in excess of one.

Each record has one attribute, called the "main object property" whose value is the name of the object represented by the record.

There are apparently no cross references.

4.7.5 Query Language

A query consists of three major phrases.

The first one consists of the single word GET, called the Major Directive, followed by the File Index, which is the name of the file being queried.

The second phrase begins with the Conditional Directive, IF. IF is followed by a sequence of one or more property/value conditions. The conditions are conjoined by the (Boolean) Logical Connectives AND and OR. Each condition consists of three parts in sequence: an elementary property (or Attribute), an Attribute-Value Operator, and a value.

The third major phrase begins with one of the three Sub-Directives, PRINT, DISPLAY, and COUNT. The remainder of the major phrase is the Sub-Directive Modifier. The Sub-Directive Modifier is either (1) the word ALL, (2) the word IT, (3) a string of Attributes, or (4) the word NOT followed by a string of attributes.

A representative query and its structure are given below:

GET AFLD IF COUNTRY = US AND RNWY-LNGTH > 7000 PRINT
AFLD- NAME RNWY- LENGTH

4.7.6 Retrieval Logic

An English paraphrase of this query is "Print the names and the lengths of the runways of all airfields in the United States which have at least one runway longer than 7000 feet."

The Major Directive, GET, identifies the query. The File Index selects a particular file for the query. IF always marks the beginning of the second major phrase. The property/value conditions define conditions which records in the selected file must meet in order for their data to be relevant. In the example, the value of the elementary property COUNTRY must be US and the value of the elementary property RNWY-LNGTH must be greater than 7000. Subdirectives define the computation to be performed and the output medium. PRINT specifies a high speed device, DISPLAY a low speed device, and COUNT directs a count of the qualifying property values in the Sub-Directive Modifier. The Sub-Directive Modifier specifies the data to be extracted. In the example, these data are the values of two elementary properties. The Sub-Directive Modifier may be the single word IT, in which case the values to be extracted are precisely those whose properties are mentioned in the second major phrase. It may also be the word ALL, which specifies that all the data in each qualifying record is to be extracted.

The search component of the retrieval logic begins by reserving a "condition indicator" for each condition in the second major phrase. If the property named in a condition is a top level one and the condition is met at least once in the record (i.e., in the master record or in at least one of the trailer records) the corresponding indicator is set. If the property named in a condition is a second-level property, the corresponding indicator will be set if and only if the condition is satisfied at least once and any other conditions with properties subordinate to the same top-level property are also met.

When all conditions have been examined, their indicators are "AND-ed" and "OR-ed" together to determine whether or not the current record qualifies.

For a description of the extraction component, assume that only PRINT or DISPLAY is used as Sub-Directive and that IT and ALL are not used.

If a top level property is mentioned in the third major phrase and not in the second, all its values (for the master and all trailer records) are extracted (printed or displayed).

If a property is mentioned in both, only the qualifying values will be extracted. If the property is top level, all of its subsumed values will be extracted. A second-level property can be printed only if its accompanying properties qualify, and only qualifying values will be printed.

4.7.7 Capability

Mathematical and logical operations using the system's data base are provided by both stored program and on-line programming techniques through the use of a query-control language. Control of the data, computer programs and equipment is achieved

through on-line interpretive execution of the COLINGO Control Language (CCL) instructions entered by cards or from an inquiry console typewriter. The design includes a Basic Program Set which provides programs for rapid and convenient data file update, data file addition, and stored program addition. This design allows sequential expansion of the Basic Program Set and the addition of special purpose and convenience features. On-line file generation and on-line programming in COLINGO are an integral part of the system design. Special data maintenance and verification routines as well as automated system dictionary and vocabulary preparations are also included in the system. These features will permit evolutionary growth of the system and provide the capability to produce special purpose operational programs with a minimum of programming time.

In order for a user to quickly obtain results from a generated data file, a combined query and programming language was provided to manipulate the data base both logically and mathematically. Additionally, a general-purpose output program was provided to present results to the user in formats of his choosing.

A simple allocation scheme for introducing special-purpose programs into the COLINGO System was provided. This scheme allowed a high degree of communication between the COLINGO Basic Program (BPS) and the special-purpose programs.

The COLINGO Control Language was structured in a way that provided a convenient query language to complement a more powerful language useful in many programming applications. The combined language was designed to divorce the user as far as possible from the computer equipment configuration and to remove the usual core-size limitations on the operational problem being programmed. This removal was accomplished by the process of chaining operational statements together and allocating them to disk, and having the chain sequentially called in by a small in-core Executive routine.

An essential characteristic of COLINGO is its ability to map many physical data formats into its own format in a short time and then to query the resultant formatted data with facility. As a consequence, COLINGO is a highly flexible information retrieval system.

The COLINGO design achieved a large degree of data and program independence by not referring to data fields directly, but instead through a data describing dictionary. This dictionary was created according to COBOL procedures, but achieved the important advantage of being modifiable on-line by the operator.

Simple data generation and verification programs were included in COLINGO to generate a large variety of card and tape data files from external sources, and in addition, the CCL itself was given the power to manipulate and change its own data base as to content and format. Thus, not only could the COLINGO System accept a wide variety of formats, it could also process and output that same data in a variety of different formats.

4.7.8 Comparison with Reliability Central Data Management Subsystem

COLINGO represents a significant advance in the state of the art of dealing with large amounts of data on a small computer. The COLINGO design has a number of features which are similar to the objectives of RCDMS, but the designs differ widely in the relationship between the system and the data base. Some of these differences are discussed below:

(1) Because most of its data is on magnetic tape, COLINGO is more file-oriented than data-oriented. If a query cannot be satisfied by the limited QUIC files on disc, the system must turn to magnetic tape for its answer. COLINGO can search several magnetic tape files in behalf of one query, but these files must be searched serially without a logical

connection between them because there are no cross references between files (or within files) of the system. RCDMS is not restricted in its access to files because it deals with the data base as a single entity. RCDMS has a consolidated directory as well as random access storage for all summary data that is normally used to answer queries. This feature permits the system to update or interrogate files in parallel, and to link files according to logical criteria.

(2) The search strategy of COLINGO is to scan in a linear fashion the serially organized magnetic tape files or data portions drawn from the QUIC files. This means that all records to be searched must be brought into core memory, and each record must be compared with the criteria of the query. RCDMS contains data indexing in its directories so that it can use a search strategy which manipulates the directory rather than scanning the data. Once the pertinent portion of the directory has been brought into core memory, the RCDMS search package can determine under certain circumstances, the exact addresses of all the data elements which satisfy the query. In other instances, a small subset of the file is retrieved and those retrieved records must be compared with the criteria of the query, as in COLINGO. By accessing only the pertinent data elements, needless accesses are eliminated.

(3) The COLINGO search logic does not permit the use of the NOT operator. In addition, functional operators require two passes through the file, since a serial search strategy does not provide the required communication between files or between records in a file. This will not be necessary with the RCDMS design.

(4) COLINGO is able to nest up to fourteen data items within another data item, but these must be at a single level of nesting. The data anticipated for the Reliability Central will require a more complex data structure. RCDMS has made provision for an arbitrary number of reliability test records to be associated together at any level in the data hierarchy.

The features in the COLINGO design which are comparable to the Reliability Central objectives and RCDMS design are as follows:

(1) COLINGO is one of the few systems surveyed that utilizes a flexible and wide-reaching data description language. It uses a language similar to the COBOL "picture." A similar degree of user and system flexibility is obtained with the data description language designed for RCDMS, which permits almost any data structure to be specified, and which will offer convenient data formats for the user who is not computer-oriented.

(2) COLINGO offers a comprehensive capability for file generation and maintenance.

(3) The COLINGO Control Language is able to specify a wide variety of jobs and queries. The RCDMS Job Specification Language has a similar capability for specifying jobs. The RCDMS query language has not been designed, but the system search package is designed to accommodate a query language which:

(a) uses the Boolean operators of AND OR and NOT,

(b) uses such relational operators as =, >, <, and ranges,

(c) permits arbitrary nested conditions,

(d) searches at any level of the data structure.

4.8 LANGUAGE USED TO COMMUNICATE INFORMATION SYSTEM DESIGN (LUCID)

4.8.1 Objectives

LUCID (Language Used to Communicate Information System Design) is a general-purpose programming system under development by the System Development Corporation for ARPA. Its primary objective is to provide a tool for the design of data management systems. The LUCID view is that any computer-based command and control system is essentially a data management system and that the problems of the data base are central to the problems of design. LUCID attempts to provide the system designer with a tool that will help solve the problem of system data content and structure earlier in the design process. It does this by providing both a language in which design of data management systems can be expressed, and a programming system to convert this expression of design automatically into something that can be run on a computer.

4.8.2 Implementation

The feasibility model of LUCID is written in J-2 Jovial for the AN/FSQ-32 with tapes and drum. It does not operate under the Q-32 monitor system because of the space limitations imposed. Assembly language was used for the I/O subsystem but is believed to be on the order of 5 percent of the total system.

Extensive use was made of previous work in developing components of the LUCID prototype which is currently running. The prototype which was not written with any expectation of moving it to another machine depends heavily on the drums in the Q-32 system used for the directory and related tables. In addition, the program size (including communication tables and data) is limited at present to 16K. If the system were implemented on another (smaller) machine, it would require re-organization.

A separate (non-integrated) Utility system was designed and built for the programs and libraries which make up the prototype LUCID system. An initial version consisting of the file description language interpreter, a data load function and a query capability has been implemented to demonstrate the feasibility of the concept. The full LUCID design is being reviewed for implementation under a planned SDC time-sharing system.

The Query System implemented in the feasibility model operates on a single file (termed data base), and is activated from a console typewriter. Two subcommands have been defined: PRINT (enumerate on the typewriter), and LIST (creating a tape file for off-line listing). Only numeric information is obtained, and can be specified by any arbitrary Boolean and relational expression. In a conversation with the system designer, it was determined that this facility would be expanded to include alphanumerics in the planned full implementation.

4.8.3 Components and Data Structure

At present, the LUCID System consists of the three following major functional components, and their associated system programs:

(1) A language for describing a data file.

(2) A language to describe inputs into the data file.

(3) A query language to query the data base and retrieve data from it.

The language processing and system programs are developed from and utilize the DODDAC Data Base Load (DBL) programs, and the GENDARME System of Data Processing procedures (tape oriented).

4.8.3.1 Data Base Description. In its current version, LUCID accepts a definition of a data base.* The maximum number of terms currently contemplated for a processable data base is approximately 100. The data base description can cope with the following types of items:

(1) Name, meaning an item containing an alphanumeric string.

(2) Descriptors for data whose values are in integer form.

(3) Descriptors for items whose values are in floating point form.

(4) Descriptors for categories of items where a name for a class of items is followed by the word CATEGORIES and descriptors of items composing that class e.g.,

TITLE CATEGORIES MANAGER, CLERK.

(5) Descriptors for strings of items; categories may be imbedded in the string e.g.,

STRING SET CONTAINING
SCHOOL NAME
YEAR MAX. 2000
DEGREE CATEGORIES PHD. MA. BA.

(6) Descriptors of items whose values are to be included in the data base if the given stated logical condition is satisfied.

*"Data base" is the LUCID term for file.

The data base is read in and Data Base Table directories are created to be used subsequently when data input to the data base is submitted. Essentially, the data base description can be viewed as description of a file entry, where each data entry in the file will be described by the terms found in the data base description.

4.8.3.2 Input. Input of data into the data base is carried out in two steps:

First, a description of the data to be input and of the format in which it is keypunched is read in and processed. The input command may specify the data description in the following way:

FORM MEANS SYMBOLIC INPUT

Option 1 { CARD SET IDENTITY n
Option 2 { DATA SET TERMINATOR END

Possible Data Forms { PACKED or
SEQUENCED or
NUMBERED

followed by: Names of Data to be entered, numbered where sequenced input is not specified (i.e., sequenced in the sequence of the original data base description).

The LUCID translator programs operate on this input, producing the following results:

(1) Master List Directory to OPAQUE Tables.

(2) Table describing the Data Base.

(3) Table of Symbolic Inputs.

The data itself can then be read under a command, such as

MAKE (PERSONNEL DIRECTORY) FROM (FORM)

The term MAKE refers to a load file which contains the program to load data.

The system will read the data of one entry at a time, constructing a concordance dictionary for all terms described in the input and for all data values input. The directory is organized as follows:

(1) COIL Table, containing an entry for each term of the data base for which input has been defined. Each entry in the COIL Table points to entries in:

(2) JUST Table. JUST Table contains for each entry in COIL Table a list of Values which exist in the data for that term. Each entry in the JUST Table points to:

(3) ALTO Table. ALTO Table contains a list of addresses for each value in the JUST Table, pointing to data entries in which the value occurs.

Thus, a complete concordance (cross-reference) of all data values, terms, and entries is developed, which lends itself to a rapid search for query purposes. However, a directory several times the size of the data results.

It is planned to make the concordance optional, so that complete cross-referencing need not result where it is known not to be required.

At present the LUCID concepts and directory structure cannot recursively imbed structures in the data base. Currently processes for an ORGANIZE COMMAND are being developed, which will result in organizing the completely cross-referenced data base into two structural levels.

(1) Files, which may be recursive e.g., country, state within country, city within state.

(2) Tables — e.g., building specification table.

4.8.3.3 Query. The last major component of the LUCID system is QUERY.

The query permits searching the data base in response to a query statement containing Boolean and relational operators. Requests involving summation and development of simple arithmetic mean are also allowed. The following major query components are provided:

ITEMS — System will type out list of items available in the data base

e.g., NAME
SALARY
LENGTH OF SERVICE
ETC.

SHOW — System will show list of values available in data base for a given data base term.

PRINT — Print, followed by Boolean expression will result in the retrieval and type out of data satisfying the Boolean expression. Where a lengthy output is anticipated, the command should be LIST; the output will be prepared for printing on high speed printer.

COUNT — The system will develop and present a count of occurrences of a condition specified in query.

4.8.4 Capability and Use

LUCID has a vocabulary of over 200 words which correspond to particular operations. The user may change the word corresponding to any function very easily on-line through an equivalence function built into the system. There are also a number of noise words which have no meaning and are ignored. These noise words help to make the LUCID input more readable.

The LUCID translator converts the input text into a binary tabular form. These tables, OPAQUE tables (Operational Parameters to Activate and Qualify Users' Expressions) are examined by an interpreter which then calls in the necessary GENDARME subroutines and executes them. Some of the OPAQUE tables are simple and are merely abstracts of other tables made in the interest of more efficient operation.

Currently, any change in the data base description (such as the addition or deletion of descriptors) requires complete reentry and reprocessing of the data base description.

User analytical functions to be executed by user programs referring to the data base are not provided as part of the LUCID system, nor is a dynamically changeable user program and job (sequences of programs) library.

The initial version of a system design is run in an interpretive mode. It is being used for demonstration as a base for future development. The inefficiencies of this mode are offered to allow for convenient debugging and experimentation. The following facilities are thought of as possible in the future: when the system design is stabilized, a

generation process can be undertaken to produce automatically a set of non-interpretive programs that perform the same functions that the interpretive system had performed. This second mode would operate more rapidly, possibly to meet some real-time constraint that the interpretive system might not be able to meet. The user would continue to use the interpretive version for development purposes, and would never modify the generated version. New models are introduced by regenerating. This device would be used in an effort to obtain the flexibility inherent in interpretive systems, combined with the production efficiency and non-interpretive versions. Another possibility contemplated is the development of generators for several different machines, producing a version of the system for a particular machine after simulating the machine on the original computer in the interpretive mode.

4.8.5 Comparison with Reliability Central Data Management Subsystem

LUCID and RCDMS are similar in their approach to describing a data base in a language natural to the data which the data base is to contain and in their approach to data input description and data input. RCDMS, however, provides additional facilities for operating many user programs either individually or strung into jobs predefined in RCDMS library. This facility is not currently provided in the LUCID system.

The points of functional similarity exist in the following areas:

(1) Description of data base.

(2) Description of inputs into the data base.

(3) Facilities for querying data base.

The differences are found in the following areas:

(1) RCDMS permits updating of data base description without having to re-enter and re-process the whole data base description.

(2) RCDMS provides system programs and jobs which permit the user to prepare and execute analytical and other problem-oriented programs using data from the data base, without the need for the user to be concerned with the detailed mechanism of data retrieval.

(3) RCDMS views the data base as containing data structures defined by many users for many processes and is not limited to a single file. Thus, files are defined containing arbitrary number of records, within which other files may be defined recursively, as well as statements and items. At the same time, the RCDMS directory system permits retrieval across the data base as a whole. Indexing by data values is optional, permitting dynamic maintenance of tradeoffs between search speed and storage requirements.

(4) The size of data base description in RCDMS (i.e., number of terms) and of the data base itself is limited by the overall storage capacity of the EDP system in which it is implemented, rather than by the storage capacity of the processor available.

4.9 ADVANCED MANAGEMENT SYSTEM (ADAM)

4.9.1 Objectives

ADAM (Advanced Data Management) is a system being developed by MITRE Corporation under sponsorship of USAF ESD. The primary purpose of ADAM was to provide a test-bed for command and control system designs in the System Design Laboratory at the MITRE Corporation. The ADAM system was to accomplish this by providing a software environment in which the languages and the system description that specifies what the simulated system shall do are considered as data by ADAM and can be changed with the ease in which the data base of the modeled system can be changed.

The basic ADAM goal is to provide the ability to investigate the behavior of proposed system designs by creating files for that system, processing queries against those files, and running users' programs that use the files and the result of file queries. More specifically, the system must provide the following:

(1) Data input in a wide variety of formats and content. In general, this will mean that after a description of data in any machine sensible form has been entered in the computer, the data will be acceptable for system processing.

(2) Comprehensive facility for updating the data base. This facility includes the creation, augmentation, alteration and reorganization of the files on a dynamic basis.

(3) Reorganization of information in reports (and query responses) according to preplanned specification, including the preparation of summaries and sorting on multiple keys. If necessary, the format in which reports (or query responses) are displayed can

be specified by an operator and can be varied by him on a dynamic basis.

(4) Facility for incorporation of "model specific" code (i.e., routines and/or models outside of ADAM which pertain specifically to a given detailed calculation or perform a specialized operational function). Provision should be made for programming these specific routines in a higher order language. The program design must permit the concurrent checkout of such routines without compromising the integrity of the data base or interfering unduly with the operation of the rest of the system.

(5) Easy communication with the computer. The use of abbreviations and synonyms would be extensive in order to reduce difficulty of communications with the computer. "Standard" queries or standard "phrases" in queries and standard formats should be used where applicable.

(6) Capability to provide the data required for the specific models and computations for internal use, or for external off-line use.

(7) Provide rapid response to shifting emphasis and priority, so that both the hardware and and software systems will be used effectively in satisfying current requirements.

(8) Capability to optimize computer time utilized by batching outputs so as to allow sorting the data base prior to an output exercise mode.

The implications of the above are that, in general, references to size, content, and format of files, messages, and displays must be isolated from the rigidly coded "Model Specific" programs. These characteristics must be part of the data base itself. With this information in the data base ADAM becomes generative and interpretive, thus gaining in generality.

4.9.2 Implementation

ADAM is being implemented for the IBM 7030 computer (Stretch) with a 65,000 word core memory (64 + 8 bit word length); two type 353 (modified 1301) disc units with a capacity of two million words, DDI Display consoles with typewriter, 64 computer-selected background transparencies, a CRT display, and a light pencil; a Stromberg Carlson printer (3000 characters/minute); and a type 7050 Multiplexer-computer.

4.9.3 Components

There are three elements of the processing task: the data, the equipment, and the processes. The data are the raw material with which the processing task must work; the equipment is the tools (processors, storage, and I/O devices) to be used in that work; and the processes (file maintenance, information retrieval, and model specific programs) determine the way in which the tools are used on the raw material.

Allocation and control of these elements reside in the ADAM system.

ADAM can be divided into three asynchronously operating sections: Input, Thruput, and Output. A diagram of the structure is given in Figure 4-2.

The Task Section is the program that responds to an (input) message. The Task is supported by a group of system programs called Thruput Ground while all three sections are supported by a group of system programs called Ground.

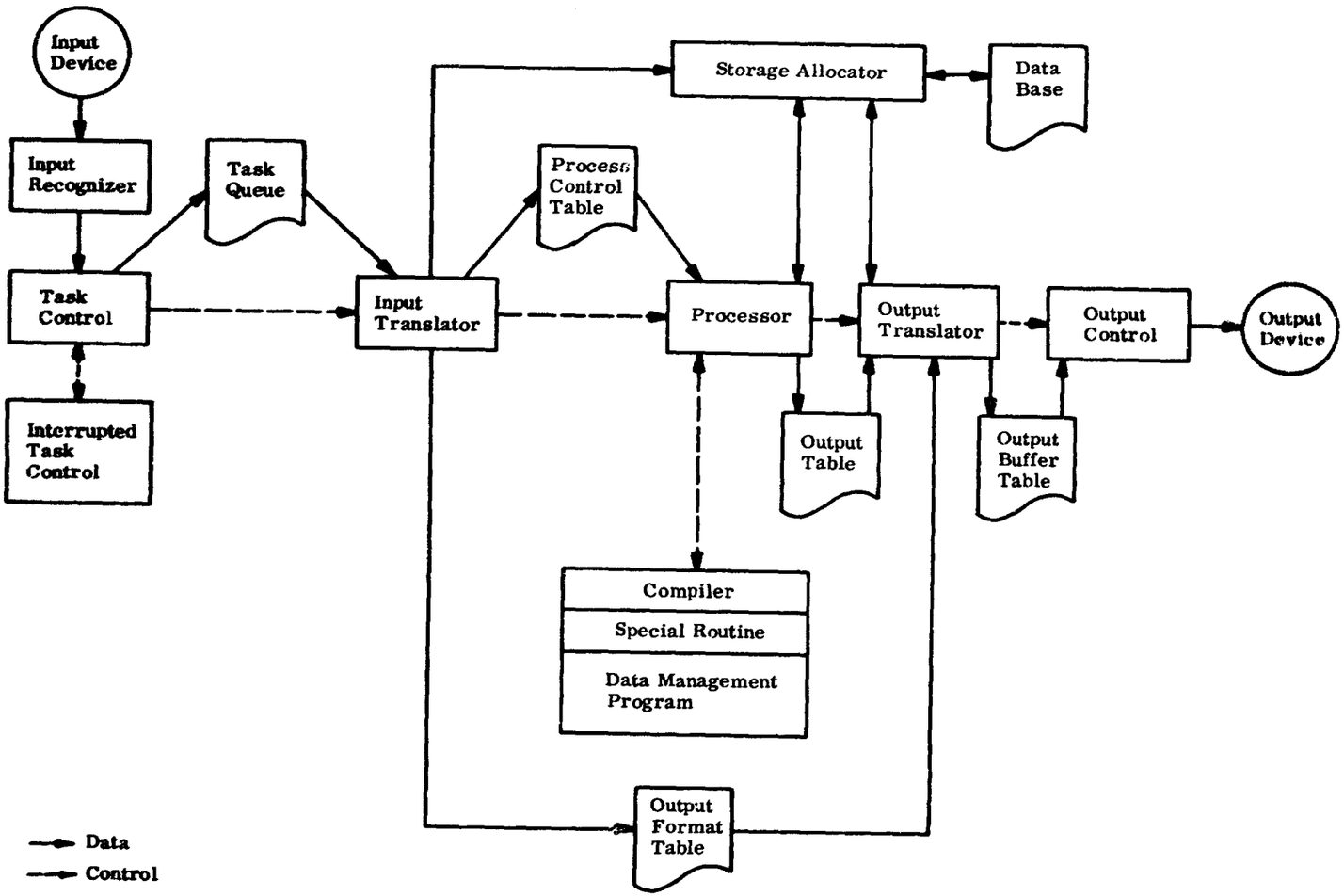


Figure 4-2. ADAM Components

Input, Output, Thruput Ground, and Ground are in the system at all times. Allocation of primary storage to input and output is not dynamic but a pseudo-infinite buffer called a STREAM is used which automatically moves data to secondary storage when necessary.

Ground consists of:

(1) Secondary Storage Manager, which allocates secondary storage and performs I/O.

(2) Debugging tools.

(3) STREAM Controller.

The Input section consists of four parts; Input Control, Input Recognition, Task Control, and Interrupted Task Control.

(1) Input Control handles all inputs to the system. It is essentially a device-oriented program which must be capable of handling input queries, input messages, updating messages, file descriptions, and control system input from teletype input, cards, torn tape, magnetic tape, or other machine sensible forms. Inputs to the system are converted, if necessary, by Input Control and presented to other input programs in a standard system code.

(2) Input Recognition scans input using rules established for it to determine the purpose of the input. At this time, appropriate translation rules are selected for later use by the Translator. The task of further processing of this input takes into account the priority established by the user.

(3) Task Control maintains a task queue and controls sequencing of tasks in accordance with the user established priority.

(4) Interrupted Task Control saves the task which was interrupted and restores it when the interrupting task is completed. Restoration of the interrupted task may be either at the point of interruption or at the beginning of the interrupted task, as required by the user or program logic.

The Thruput section, the heart of ADAM, has six main parts. These are Input Translator, Processor, Output Translator, Storage Allocator, Compiler, and File Management programs. The Input Translator translates input messages (e.g., queries, updating messages, control inputs, format descriptions) using appropriate transformation rules and produces entries in a process control table which control the operation of the Processor.

In addition, the Translator produces a description of the output which will be produced by the Processor. The Processor interprets the entries in the process control table, directs the operation of routines from the program file or the collection of selected data from the data base, and prepares an output table.

The following elements are involved in translation and execution of source (user) language input:

(1) THRUCON — The control program

(2) SEPSCAN — The input interpreter which uses a rules table for recognizing operators in the source language and their scope.

(3) SUBSCAN — The table of substitutions.

(4) PURE — "pure" translation. It takes an input called DIAGRAM which is a diagram of the syntax of a source language, whose nodes are syntactic types and basic symbols and whose branches are allowable substitutions. Tables called STRING and LEX are generated at this stage of translation. LAP is a program for generating DIAGRAM from the syntax rules expressed as a substitution grammer (i.e., Baccus-Naur Form).

(5) TRANSPRO — Initializes OUTPUT FILE, finds generators for the entries in STRING, and generates the program table SINTAB (a pseudo-infinite table).

(6) PROCESSOR — the executive that follows the steps of SINTAB in the interpretive mode.

(7) COP — An input control program.

The Output Translator, using tables created by the Input Translator and the Processor, translates data for output from its internal form to a form acceptable to the Output Control program. Formatting is also accomplished at this time.

The Storage Allocator controls the allocation of main and secondary storage for routines and file data required by a task. Tapes, printers, storage devices, and input devices are also allocated centrally. Since programs refer to devices symbolically, the allocator can change assignments if some units are unavailable. This minimizes the dependence of any given program on the specific hardware units of any given type.

Higher language compilers are a part of ADAM and produce model specific programs compatible with the general system. The compiled programs are relocatable by hardware or software, and have all requirements for data from master files stated in symbolic form. In this way, data requirements of model specific programs are isolated from master file structure and storage location.

The File Management program is capable of generating, updating, augmenting, purging, extracting, reorganizing, and verifying system files.

The Output Control program handles output to all terminal devices. It is able to transform data presented in a standard form to the form required by the output units. In addition, this program isolates the rest of the system from the effects of slow output devices.

4.9.4 Model Specific Program Characteristics

The model specific programs operate under the executive control of ADAM. The user can dynamically set priorities for the operation of routines and for the use of the system hardware for those routines.

The addition of new model specific programs to the system does not necessitate the rewriting or recompilation of existing model specific programs. A new model specific program is added to the system in the same way that the data base is updated or a library tape is augmented. Only those characteristics of the model specific program which interact with ADAM will be discussed, since for other than these, the model specific program characteristics are internal to the routine and are not constrained in any way.

ADAM handles all storage allocation, interrupt control, and I/O assignment for each of the model specific programs.

Each model specific program, in order to be operated by ADAM, must be:

(1) Identifiable by a unique name, stored within the files used by ADAM.

(2) Compiled so that the data requirements of the program can be acted upon by ADAM, i.e., must be compatible with storage allocation as handled by ADAM.

(3) Compiled so that I/O is handled by ADAM.

(4) Relocatable, either through software or hardware. Model specific programs must be written to operate within ADAM, allowing it to handle all specific hardware references. The model specific program must consider the hardware as extended by the general program as its environment.

In addition, there must be a capability for implementing model specific programs and computational routines which use the data base in ways not initially anticipated. This means that file structures planned for optimal performance of initial models may be less than optimal for later models. Thus, "model" programs must not

require that master files have a specific and unchanging structure. Model programs must be isolated from changing characteristics. However, file structure cannot be completely independent of the processing tasks but at any time reflects a compromise most appropriate to the current state of evolution of the system.

4.9.5 Processing Control

There is a significant difference between a data processing system that operates sequentially on tasks preplanned by a human staff and one which must accept information and processing requests in an unplanned manner; i.e., not under control of the processor but rather at the direction of an external device or operator. Unplanned occurrences must be handled almost immediately as they may change the task priority and thus the processing that the system should currently undertake. For communication links, the information to be transferred may be available only for a short period of time. In addition, the control program for an on-line system must be capable of controlling the allocation of its equipment as a function of current load and current availability.

The unique devices to be shared by all programs within a system through a common control program are discs, drums, core, and some input-output channels. These devices are rarely assigned for exclusive use by a user program. Master files also fall into this category because they must be accessed by many programs and cannot be assigned for exclusive use of any one program.

Other devices and data files in the system are requested by user programs through a common control program and are assigned for the exclusive use of a program. These devices and data files can then be accessed directly by the user program.

The third category of equipment, buffered devices such as printer, displays, typewriters, and other low speed devices, are centrally controlled and are not assigned exclusively to user programs. The central input-output control program does speed matching of the devices with the internal processing and handles output separation of data to the same device.

4.9.6 Programming Languages

Two procedure-oriented programming languages, FORTRAN and DAMSEL, are available in ADAM as an initial higher order programming language capability.

Routines prepared in either of these languages require some off-line post-processing to produce some special binary code with information required to communicate with data or other routines. This binary routine is what is then inserted into the ADAM system. Routines and their communication links may also be directly prepared in the SMAC and STRAP assembly languages, and assembled off-line. Routines are all relocatable but are not all automatically segmented. They are limited by the amount of available core when they are run. All ADAM routines are kept in a file called the "routine file."

4.9.7 Data Structures

Three types of data structures are used in ADAM: Streams, Files, and Rolls.

A Stream is a pseudo-infinite buffer mentioned previously which is allocated to a task by the Thruput (or Task) Section. Movement of data in a Stream between primary and secondary storage is accomplished automatically whenever necessary.

The data base itself is in the form of ADAM files. Files are described physically by self-contained tables and changes in file format are recognized by the system, specifically by the loader/allocator called RENOVATION. The highest level

logical record in a file is called an "object" and is made up of items. Items can be of the following types:

(1) Fixed — numeric or logical.

(2) Repeating Groups — a file type structure within the object (record) which has an unlimited number of repetitions and whose elements may have an internally nested repeating group. Repeating groups may be named or unnamed.

(3) Raw Data.

(4) Query Valued — these are cross references to other objects, and functions on items or string substitutions called "Re-circulation" (Query Valued items may not be implemented in the initial system).

A Roll is a dictionary or directory that associates external names with their more concise internal representatives, usually some small integer. This integer is called the principal value or PV of the name. Synonyms for external names are effected by a many to one mapping of names to integers; i.e., all synonymous external names have the same PV. Additionally a roll will have subsidiary values or subvals which give additional information about the things named in the roll. Within a roll, PV's are unique.

The heart of ADAM data management is the ADAM file and the rolls corresponding to each file. All data objects, language specs, routines, formats, etc., are carried in files. A file is a named collection of like things. For example the "airfield" file would contain information about several specific airfields. Each airfield, e.g., Logan, Hanscom, etc., would be an object (record) in the airfield file. The various data about each object are called properties (attributes). Typical properties for the airfield file would be Location, Number of Runways, Elevation, etc. The actual data in a file are the values of these properties, e.g., Boston, 6, 2,000 feet, etc. The set of properties for an object in a file must be the

same set as for all other objects in that file. Of course, the values of these properties may change from object to object. The set of all property values for one object in a file is called an entry (record) in that file. ADAM files are structured serially by entry, the first entry being the values of the properties that occur once per file, e.g., file name, file size, data updated, etc.

Each file has associated with it, but physically separate, an object roll and a property roll. The object roll contains the names and PV's of each object in the file and subvals which give the object's location in that file. The property roll has the names and PV's of the properties in the file, and for each property, several subvals giving type, size, unit of measure, location, legal range of values, etc. Files may also use rolls to hold the alphanumeric values of some of the file's properties (e.g., red, green, up, north, or some other non-numeric value) and carry only the PV's of these values in the file itself. Many files may share such a roll. Rolls are not only used by files; the system keeps several rolls for its own internal use.

Property types are fixed-length (integer, floating, small range or low precision floating, or alphanumeric) and variable-length. The latter consists of query-valued (some processing is invoked to produce the value), raw (arbitrary string of bits such as an actual alphanumeric string as opposed to a PV standing for such a string), and repeating-group. A property of the repeating-group type is simply a collection of sub-properties, which may themselves be of repeating-group type, nested to arbitrary depth.

4.9.8 Capability and Use

The foregoing has indicated how ADAM provides considerable hardware and software support for data management system modeling. It does so by providing generalized

implementation of those data management functions which are considered to be common housekeeping functions in most systems, namely:

(1) Data base creation, maintenance, and restructuring.

(2) Data base analysis and search.

(3) Input message analysis.

(4) Output, report, and display generation.

(5) Job sequencing.

In ADAM, generalized programs for implementing these functions are written without explicit reference to data names, formats, query language, etc., which are considered the specifics of the application. Because of this, data terms, data base organizations, query language syntax, and job routines must be specified by the experimenter. But also because of this they become parameters of the system whose design can be varied and tested.

New input languages or modifications to languages are made through off-line table interpretation and assembly followed by on-line additions to the language file. Likewise, program preparation requires off-line compilation and post-processing followed by on-line additions to the routine file. File generation, processing, and problem-specific operations, on the other hand, all result from message input through on-line devices.

4.9.9 Comparison with the Reliability Central Data Management Subsystem

There are some important dissimilarities between ADAM and RCDMS in terms of design and objectives. ADAM is oriented toward the effective construction and operation of a system design in a testing environment. RCDMS is oriented

toward the efficient management of a data base for a production operation. As a consequence, ADAM tends to sacrifice some operating efficiency in order to be easily modifiable for changes in the design it is evaluating; RCDMS tends to sacrifice some flexibility in order to operate efficiently. Some of the specific differences are discussed below:

(1) ADAM does not have a capability for stringing many programs together in the form of a predefined job which can be stored for future use. The job specification function of the RCDMS design sequences programs according to user requirements, and makes provision for inserting program parameters.

(2) ADAM does not provide automatic program segmentation, with the result that routines are limited by available core space. The Local Control Program and Executive Control System of the RADC complex permit, in effect, unlimited program size by breaking programs into segments which are floatable in core storage and which can be called from random access by segment name. However, where the RCDMS capability for floatable program segmentation is used, the program so executed has to operate in interpretive mode.

(3) ADAM makes no provision for indexing of the data base by carrying the values of the file properties in its directory.

Except for named items, searching is performed by sequential scan of the file. A major feature of RCDMS is that it provides for indexing of all data to any depth of embedding in the data structure. Associated with this feature is a RCDMS search strategy package which determines, from the index alone, those data items which satisfy the criteria of a query, thereby eliminating all unnecessary accesses to peripheral storage.

On the other hand, ADAM has a great deal of conceptual similarity to RCDMS. Like ADAM, RCDMS will use highly generalized programs to perform extensive management functions on large data bases. Some of the areas common to both systems are discussed below:

(1) System control follows much the same pattern in both systems.

(2) The two systems use similar file structures. The Object and Property Rolls of ADAM perform much the same functions as the Term Encoding Table and Item Position Index of RCDMS.

(3) Both systems can handle complex data structures, including variable items and any number of repeated groups, nested to an arbitrary depth.

(4) Once the jobs have been specified, job run control is much the same in both systems (except that for ADAM, the entire job must exist in core).

(5) The concept embodied in ADAM's flexible input language translator has been adapted for use by RCDMS. Each user is able to set up his own syntactical and semantic rules for input statements which specify or define his data or his jobs. The rules are incorporated into the system in the form of action graphs, and RCDMS uses the appropriate action graph to control each input translation. However, only one standard language will be implemented and used for Reliability Central Test Operations. The syntax analysis technique has been used because it is effective, rather than to encourage a proliferation of System Languages.

SECTION V. SUMMARY OF COMPARISONS AND CONCLUSION

The purpose of this section is to show in what areas RCDMS has drawn on work already done in the field of data handling, and in what areas it has made an original contribution to data management.

This section performs the function of summarizing and placing in perspective all the design comparisons between RCDMS and the other systems. Summarizing remarks are made about each aspect of a data management system. The aspects are the same as those defined in Section II and in its comparison chart. Section V contains a final statement which synthesizes the conclusions of the survey.

5.1 PROCESSING CAPABILITY

Until the advent of these generalized systems the processing capability of a data handling system was a direct function of the specialized goals of the system. The capability in many current systems has taken on a general-purpose nature with the advent of generalized routines in several areas: the search logic used for retrieval, the maintenance of directories and of the data base, the specialized user programs that accomplish specific tasks (but which can be written in a generalized form), the rendering of output for external use, and the management functions performed by the system.

5.1.1 Analytical Functions

The power of a data handling system is greatly enhanced if the system can take the output of a query search and process it further for the user. Most systems have a limited capability in this area. The analytical functions are restricted to tasks which are specific to the mission of the system and which are called upon by specific commands in the user language. ADAM is more flexible because the analytical functions can be incorporated as part of the system being modelled by ADAM. RCDMS gives the user freedom

to incorporate his analytical functions into any part of the task he wishes performed by entering his user-specific programs with a Program Entry Request, and by entering the parameters and sequence of the programs with a Job Entry Request.

5.1.2 Search Logic

IPS, LUCID, ADAM, and RCDMS are the only systems with an unrestricted repertoire of search logic operators. COLINGO and 473L do not handle the NOT operator. COLINGO requires two passes of the search file to process functional operators, and 473L permits no nesting in the query. ASO and ACSI-MATIC omit the OR and NOT operators, as well as functional operators. RECOL omits several of the relational and functional operators.

5.1.3 Data and Directory Maintenance

All systems but RECOL maintain their directories on an automatic basis. IPS, RECOL, and 473L require the programmer to maintain the data base with special programs, but the rest of the systems maintain the data automatically. Most systems generate their files off-line. RCDMS has both off-line and on-line file generation.

5.1.4 Output Formatting

All systems have some variable means of reporting data to the user. RECOL provides merging and sorting but no specific formatting services. ACSI-MATIC and ASO have several rigid report formats that the user can select with the input language. IPS and 473L have generalized sort and formatting routines which are controlled by the query language. COLINGO has a report generator and sort routine. LUCID and ADAM provide output control, but require the user to provide his own output routines. RCDMS will initially have a generalized sort and report routine, and will eventually have a more flexible formatting routine under control of the query language.

5.1.5 Job Management Services

RCDMS makes a further unique contribution to data management systems by having not only a job management capability, but also a capability of pre-storing Job Descriptions for the user. For jobs which are repeated many times with only minor variations, the user can enter the Job Description some time before running the job. RCDMS will compare the parameters and sequencing of the Job Description with the requirements of the Program Descriptions in the Program Library Directory, thereby assuring the user that the job will be run properly. The user can also change parameters at run time, if he wishes. RCDMS jobs are to be run by means of console entered job run requests.

All systems provide an executive routine, but with a varying capability for performing job management. RECOL has virtually none of the aspects associated with job management. Several systems, such as ACSI-MATIC, ASO and IPS, use their executive to control the generalized system tasks of I/O, maintenance and retrieval, but provide no means of job specifications and job processing. Job-type sequences can be initiated by the 473L, COLINGO and, to a limited extent by LUCID user languages within the framework of the language. ADAM provides complete job run control of user specified jobs, as long as the entire job programs can fit in core memory.

5.2 USER LANGUAGES

Users require three kinds of languages to communicate with a data management system: a job specification language with which to tell the system what tasks should be performed, a query language with which to ask questions, and a data description language with which to specify how various data structures should be incorporated into the system.

5.2.1 Job Specification Language

A separate language is provided in RCDMS for job specification. Several of the systems studied have no job specification language. ACSI-MATIC, 473L, and COLINGO

perform both the job and query specification functions with a combined language. The RCDMS job language was found to be as complete and flexible as any surveyed since it incorporates the concept of the ADAM input translator which has the capability of permitting the user to design and specify his own job specification language.

5.2.2 Query Language

The query language has not been defined for RCDMS as yet, so that no comparisons can be made. The RCDMS search package, however, has been designed to accommodate a flexible query language with a comprehensive search logic, as previously described in Paragraph 5.1.2.

With the exception of the rigid ASO language, all query languages use a flexible, English-like query format. Due to a flexible input language translator, ADAM users are able to specify their own query syntax. As mentioned in the previous paragraph, this feature may not be warranted in the Reliability Central. The 473L and COLINGO query languages were found to be the ones most pertinent to RCDMS because they operate at two levels. A simplified version of the languages is available for the casual, untutored user who has a simple query. A more powerful and flexible version is available for the specialist or intermediary who has a complex query.

5.2.3 Data Description Language

Most systems have no user-oriented data description language, thereby restricting the ease with which their users can define the structure of the data. Only the more advanced systems, such as COLINGO, LUCID, and ADAM, make provision for a user-oriented data description language. As in ADAM, the RCDMS input language translator permits the user to specify his own data description language. In addition, RCDMS has a standard data description language.

5.3 FILE STRUCTURE

The concept of file structure consists of three elements: (1) a directory for describing and pointing to a general class of data such as a file, (2) data indexing for pointing to specific data items such as records and fields, and (3) the data structure itself which consists of the various formats in which the data can be stored.

5.3.1 Directory

All systems surveyed contain directories to the names and locations of files within the system. Each directory contains format tables describing the general configuration of data within each file, thereby permitting generalized routines to operate on the files. The directories are consolidated for each system, with the exception of RECOL and IPS which have a separate directory for each file. The separate directories require that each query be directed against only one file. An important feature of the RCDMS directory is that it contains only logical data addresses, which are converted into physical data addresses only when the data is to be retrieved. Less directory maintenance is required because the data segments can be moved indiscriminately from one storage level and location to another without having to alter the logical structure of the directory. This feature is mandatory for the Reliability Central because of the anticipated growth and changing structure of the data base during the future life of the Reliability Central.

5.3.2 Data Indexing

RCDMS and ACSI-MATIC are the only systems that provide deep indexing of data. Some of the systems, such as RECOL and IPS, do not have indexing because they perform a serial search of magnetic tape, and must scan the data in each record. Several of the more advanced systems, such as 473L, COLINGO, LUCID and ADAM, keep at least part of their data base on disc, but make only partial use of data indexing. This means the full

potential of random access is not exploited because portions of the data must still be searched serially after being brought in from disc. These systems have no index to point directly to specific data sets required in response to queries. The ASO system solves part of this problem by using the linked list method of indexing, where data records with like attributes are linked in a chain by means of link addresses. The linked list method reduces the number of accesses to random access storage compared to a serial search, but it has two disadvantages. First, needless accesses are still made during a search to those data records in the chain which do not meet all the query criteria. Second, file updating for deletions causes extra accesses to be made in order to maintain the links in the list structure.

The RCDMS design makes a contribution to data management systems by incorporating an efficient capability for indexing the data base in depth similar to the scheme for ACSI-MATIC. RCDMS accomplishes this indexing by taking the link addresses out of the data elements, in effect, and placing them in a consolidated list in the directory. The consolidated list permits the use of techniques which help to diminish the two disadvantages of the linked-list structure approach. First, no needless accesses are made to data storage because the consolidated lists can be manipulated to produce the addresses of only those data records which satisfy the query criteria. Second, the accesses required for index maintenance are greatly reduced because the changes can be made in a consolidated list rather than in each individual data record. Data indexing increases the storage requirement of a system, depending on how deeply into the structure the data is indexed. However, an indexing code is being designed into the RCDMS directory which seeks to minimize the storage requirement for the indices.

5.3.3 Data Structure

Most of the systems surveyed have a restricted data structure. RECOL is the most restricted, permitting only fixed length fields. 473L and ASO have no nesting of repeated groups and no cross referencing of records between files or within files. COLINGO permits one level of nesting but restricts the nested group to fourteen items, and it has no cross referencing. ACSI-MATIC and IPS permit extensive nesting with an unlimited number of items in each group, but they have no cross referencing. ADAM permits deep nesting of data, but no cross referencing.

It is anticipated that the data to be absorbed by the Reliability Central will be extremely varied and will have many interrelationships. Consequently, RCDMS has been designed to create a data structure with an arbitrary depth of nesting, an unlimited number of nested items, fixed, variable or optional data items, and cross references between data items.

5.4 PROGRAMMER INTERFACE

The data management system must be responsive to the needs of programmers in order to remain adaptable and efficient. These needs are best met by having an effective procedural language for creating system programs, by maintaining a useful library of programs and macro routines, and by providing an effective directory in the working system to the programs which can be drawn from the library for performing interim tasks or complete jobs.

5.4.1 Procedural Language

The more powerful systems (which also tend to be the more job-oriented systems) provide their users with a procedural language and attendant compilers for constructing special user programs. The programs can be absorbed into the system so as to become a

part of specific jobs. Examples are the FORTRAN and DAMSEL compilers for ADAM as well as the JOVIAL and CS-1 compilers planned for RCDMS by RADC. IPS has a special procedural language oriented to data maintenance routines. Systems with a lesser job orientation, such as ACSI-MATIC and 473L, provide their own special assembly language for a more restricted repertoire of user programs.

5.4.2 Program and Macro Library

The systems with a program sequencing capability tend to have their own library of programs which can be called by means of the user language, via executive control. Most of the programs appear to involve query or job maintenance tasks rather than special user-oriented tasks. Those systems with little or no program sequencing capability tend to rely on the macro instructions contained in their compilers for program flexibility. The systems with their own library have the distinct advantage of being able to add modular program increments to the capability of the system.

5.4.3 System Program Library Directory

A program library directory exists in all but one of the systems that form jobs by sequencing programs. The library directory is an efficient means of locating a certain program with the least amount of searching. Only IPS scans its program tape to locate a program. RCDMS uses its Program Library Directory for a special purpose. The RCDMS directory contains a complete specification of the parameter requirements for each program. Thus, RCDMS is able to help the user by checking the parameters of each job description against the program requirements well before run time. The corollary of this provision is the ability to specify input-output parameter values at job-run-time, permitting construction of user jobs from a series of closed subroutines which are parameterized.

5.5 RESPONSE TIME

Data management systems usually achieve their greatest usefulness by being as immediately responsive to user demands as possible, within the range of reasonable cost. By responding rapidly, the system not only gives greater satisfaction to each user, but it also can take care of more users. Console access is the goal usually desired by system users, but several factors influence the capability of a system to respond fast enough for console access. These factors are: the access speed of the device on which the data, directories and programs are stored; the effectiveness of the search strategy for retrieving data; the ability to grant priorities to certain requests; and the ability to store frequently accessed data on the storage device with the fastest access speed.

5.5.1 Storage Device

RECOL and IPS are the only systems restricted solely to the use of magnetic tape. COLINGO stores its programs and some of its data on disc but most of the data is stored on magnetic tape. All the other systems surveyed use disc for both programs and data. The storage configuration for RCDMS has not been determined fully as yet, but present plans are as follows: an auxiliary system core and high speed drum will be used for storing programs, operating on directories and high priority data; disc will be used for frequently-accessed data; and magnetic tape will be used for infrequently- accessed data and for back-up data, until such time as a magnetic card system becomes available.

5.5.2 Search Strategy

The large majority of systems use a linear search strategy through a serial file by means of a single, generalized search routine. RECOL and IPS are required to use this strategy because they use only magnetic tape, which is inherently serial. 473L, COLINGO, LUCID, and ADAM employ random access storage devices for at least part of their data, but

they have no indexing by data values. Consequently, once the directory has been used to select the file to be searched, the file must be searched serially. The ASO uses a faster strategy, called the list structure method, which searches through chains of linked data elements with similar attributes. The list structure method decreases response time, but increases maintenance time and storage requirements. RCDMS uses an expanded version of the ACSI-MATIC search strategy. It searches consolidated index lists in its directory and determines the address of only those data records which meet the criteria of the query. The strategy eliminates many accesses to storage by retrieving only pertinent data. Systems with data indexing require more file maintenance than those without indexing, but RCDMS reduces this maintenance burden compared to the ASO and ACSI-MATIC systems by keeping the index data in a consolidated list, and by establishing a distinction between the logical structure and physical storage of data, and by abstracting links from the data base to its directory.

5.5.3 Priority Processing

The degree of provisions for priority processing varies widely in the systems surveyed. Since IPS is designed to operate as a subsystem in an operating environment, the environment can supply the priority service. The ASO system has two levels of priority. ADAM has a complete capability for recognizing and queuing inputs according to priority. A similar capability is planned for the ultimate RCDMS design, but may not be implemented for the Test Operation.

5.5.4 Levels of Accessibility

All but the more advanced systems have a single level of accessibility to data -- either disc or magnetic tape. COLINGO has two levels: disc for programs and high priority data, and magnetic tape for the rest of the data. In the ADAM system the user can define what he wants on either disc or magnetic tape. RCDMS will have several levels of accessibility to data, as described in Paragraph 5.5.1. An auxiliary system core will be the

highest level. The next level will be a high speed drum. A third level will be disc, followed by a fourth level on magnetic tape.

An unusual feature of RCDMS is that it will attempt to control the allocation of data to the various levels of accessibility. After the initial allocation, RCDMS will keep statistics on the access frequencies to various segments of data, and will calculate the level best suited for each segment. When a segment is to be stored in the RADC complex, an indication of the desired store level (from the viewpoint of RCDMS) will be passed along to the Local Control and Executive Control Programs for their use in allocating storage to the data (from the viewpoint of the entire complex).

5.6 CONCLUSIONS

(1) The RCDMS design has leaned heavily on the advances in the state of the art already achieved in the field of data retrieval and data management.

(2) The basic contribution made by RCDMS is that it has drawn together into one system many major aspects currently associated with data management. This consolidation of system features should make RCDMS more responsive to user needs and more of a general-purpose production system than the other systems surveyed.

(3) The RCDMS design helps to divorce the user from the complexities of the system. RCDMS deals with the data base more as a single entity than as a collection of files. By permitting nesting and cross referencing of data, by indexing data automatically, by keeping statistics on data usage and by providing the user with flexible and easily-used languages, the RCDMS design lets the user alter his data easily when his needs change. More rigid systems require

the user to plan ahead more carefully, and to become more familiar with the details of the system so he will be sure his plans can be implemented.

(4) Several original contributions to data management systems are contained in the RCDMS design:

(a) RCDMS makes full use of random access potential by keeping efficiently stored indexing data in the directory and by using an optimized search strategy which accesses only pertinent data from stores available in the computer systems on which it is implemented. The depth of indexing is at the user's option.

(b) RCDMS separates the logical addresses of data from the physical addresses so that minimal directory maintenance is required when data is moved from one physical location to another, and so that actual data movement is localized when the logical structure of the data base is modified.

(c) Of all the systems surveyed, RCDMS has the most flexible and user-oriented method for having jobs specified by the user. Further, the jobs can be pre-defined and stored before run time. The program library directory contains sufficient parametric data to permit automatic linking of programs as specified by the job description.

(d) RCDMS will keep track of and analyze access frequencies to various kinds of data, and will automatically notify operating personnel of data structure changes which might be more efficient. Eventually it is hoped that these changes can be made

automatically within the system. A further by-product of the statistical analysis is that RCDMS will automatically determine the level of accessibility best suited to a segment of data and will transmit this information to the RADC Local Control and Executive Control Programs for their discretionary use when the data segment is to be stored.

BIBLIOGRAPHY

RECOL

Climenson, W.D. — "RECOL - A Retrieval command language," Communs. of ACM 6(3): pp. 117-22 (March, 1963)

ACSI-MATIC

Colilla, R.A.; Sams, B.A. — "Information structure for processing and retrieving," Communs. of ACM (Jan., 1962)

Gurk, H.M.; Minker, J. — "The Design and simulation of an information processing system," J. of ACM 8(2): pp. 260-70 (Apr., 1961)

Miller, L.; Minker, J.; Reed, W.G.; and Shindle, W.E. — "A Multi-level file structure for information processing," Proc. Western Joint Computer Conf., San Francisco (NJCC No. 17): pp. 53-9 (May 1960)

---------- — ACSI-MATIC user's manual, IR-62-2, (Jan., 1962) CONFIDENTIAL

---------- — ACSI-MATIC input message format, SR-61-4, (July, 1961) CONFIDENTIAL

---------- — ACSI-MATIC file structure, a reference manual for programmers, IR-60-2, (Nov , 1962) CONFIDENTIAL

ASO

Perry, B — "Technical report on the ASO executive routine," Report 63-14 (AD 293106) prepared at U. of Pennsylvania-Moore School

Zimmerman, B., et al. — "Naval Aviation Supply Office inventory retrieval system," Mgt Science 10(3): pp. 421-28 (April, 1964)

Prywes, N.S.; Miller, D.H. — "An Executive program for multiple consoles," Report 64-13 (March, 1964) prepared at U. of Pennsylvania-Moore School

Landauer, W.I. — "The Tree as stratagem for automatic information handling," Report 63-15 (1962) prepared at U. of Pennsylvania-Moore School

Prywes, N.S.; Gray, H.G. — "The multi-list system of real-time storage and retrieval," Proc. IFIP Congress '62, Munich, Germany (Aug. 27-Sept. 1, 1962) ("Information processing 1962"): pp. 273-78 (1963)

INFORMATION PROCESSING SYSTEM

---------- "Preliminary functional description of the CINCPAC OPCON information processing system," IBM FSD NAVCOSSACT DRG 20 (Nov. 10, 1962)

---------- Advanced programming developments -- A Survey. Bedford, Mass., Directorate of Computers, USAF ESD (Sep., 1964) pp. 18-20

---------- User's manual, IPS Phase II NAVCOSSACT No.123; June 15, 1964 (Rev. Dec., 1964)

---------- Preliminary functional design (PFD), IPS Phase II NAVCOSSACT No. 66; (Oct. 24, 1963)

Foster, D.C. "The Information Processing for the AN/FYK-1(V) data processing set," Proc. Second Congress on Info. System Sciences: pp. 97-126 (Nov. 1964)

473L

Barlow, A E.; Cease, D.R. "HQ, USAF Command and Control System query language," Proc. Second Congress on Info. System Sciences: pp. 47-95 (Nov. 1964)

---------- Advanced programming developments -- A Survey. Bedford, Mass. Directorate of Computers, USAF ESD, Sep. 1964; pp. 21-3

Rawdon, W.K. "Query languages in data retrieval systems," TM 3727: pp. 29-31 (Sept. 1963)

Heistand, R.E. "An Executive system implemented as a finite-state automation," Communs. of ACM 7(11): pp. 669-77 (Nov. 1964)

COLINGO

Spitzer, J.F.; Robertson, J.G.; and Neuse, D.H "The COLINGO system design philosophy," Proc. Second Congress on Info, System Sciences: pp. 1-45 (Nov. 1964)

Robertson, J.G.; and Spitzer, J.F. "USSTRICOM COLINGO "D" system description," Report W-06821 (Jan. 22, 1964)

Croke, J.J.; Rawdon, W.K. "Query languages in data retrieval systems," Report TM-3727: pp. 22-5 (Sept. 15, 1963)

---------- Advanced programming developments -- A Survey. Bedford, Mass. Directorate of Computers, USAF ESD, Sept, 1964; pp. 15-18

LUCID

Franks, E. — The LUCID system of automatic programming directly from data processing system design specifications. SDC FN-6797/000/00; Aug. 9, 1962

Franks, E. — System design specifications for LUCID Phase I. SDC TM-1749/000/00 (series)

Franks, E. — "LUCID," Proc. of Symp. on Dev. & Mgt of a Computer - Centered Data Base, (System Dev. Corp.): pp. 87-96 (Jan. 6, 1964)

---------- — Advanced programming developments -- A Survey. Bedford, Mass. Directorate of Computers, USAF ESD, (Sept. 1964)

ADAM

Burrows, J.H. — "Automated data management (ADAM)," Proc. of Symp. on Dev. & Mgt of a Computer-Centered Data Base, (System Dev. Corp.): pp. 63-86 (Jan. 6, 1964)

---------- — Advanced programming developments -- A Survey. Bedford, Mass. Directorate of Computers, USAF ESD, Sept. 1964; pp. 53-7

Sable, J.D — "ADAM (Advanced DAta Mgt) project," (Jan 27, 1964) prepared as AUERBACH Corp. trip report INTERNAL Report